PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 3:
G06F 7/38; G11C 8/00

A1

(11) International Publication Number: WO 80/01423

(43) International Publication Date: 10 July 1980 (10.07.80)

(21) International Application Number: PCT/US79/01045

(22) International Filing Date: 6 December 1979 (06.12.79)

(31) Priority Application Number: 974,426

(32) Priority Date: 29 December 1978 (29.12.78)

(33) Priority Country: US

(71) Applicant: WESTERN ELECTRIC COMPANY INC. [US/US]; 222 Broadway, New York, NY 10038 (US).

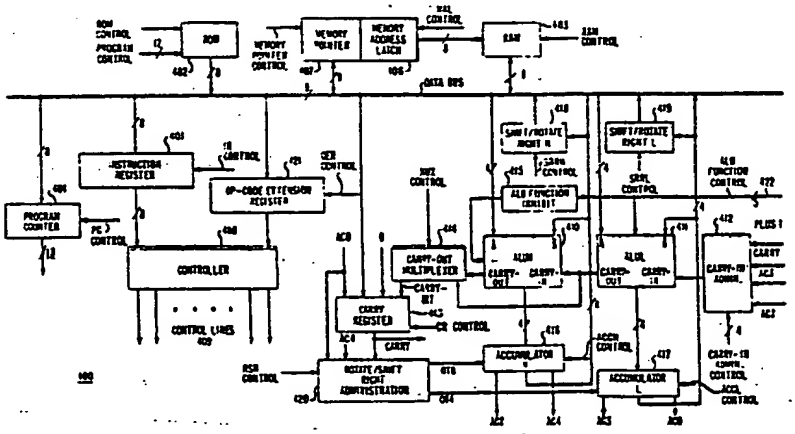
(72) Inventors: HUANG, Victor, Kuo-Liang; 2246 Jersey Avenue, Scotch Plains, NJ 07090 (US). RUTH, Richard, Lloyd; 142 Morris Avenue, Summit, NJ 07901 (US).

(74) Agents: SPENCER, Reuben et al.; Post Office Box 901, Princeton, NJ 08540 (US).

(81) Designated States: CH, DE, GB, JP, NL, SE.

Published
With international search report
With amended claims

(54) Title: DATA PROCESSING APPARATUS HAVING OPCODE EXTENSION REGISTER



(57) Abstract

A Central Processing Unit (CPU) includes a hardware op-code extending register (421) for storing a code for programmable selection of optional Central Processing Unit features which modify processor operations defined by the op-code in each instruction. A control section in the Central Processing Unit decodes both the op-code of a current instruction and the code in the op-code extending register (421), effectively combining the two to form an extended op-code capable of defining a larger set of processor operations than the op-code carried in each instruction. The code in the op-code extending register (421) is changed only when the Central Processing Unit executes an instruction for transferring a new code into op-code extending register (421). Thus the code in op-code extending register (421) can remain stationary over many instruction cycles. A central Processing Unit architecture having an op-code extending register (421) permits the set of processor operations for the Central Processing Unit to be expanded without increasing the instruction length. The advantages provided are reduced memory overhead for program storage and increased processing efficiency in data processing systems having a small word size.

BNSDOCID: <WO___8001423A1_I_>

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BR	Brazil
CF	Central African Republic
CG	Congo
CH	Switzerland
CM	Cameroon
DE	Germany, Federal Republic of
DK	Denmark
FR	France
GA	Gabon
GB	United Kingdom
HU	Hungary
JP	Japan
KP	Democratic People's Republic of Korea
LI	Liechtenstein
LU	Luxembourg
MC	Monaco
MG	Madagascar
MW	Malawi
NL	Netherlands
NO	Norway
RO	Romania
SE	Sweden
SN	Senegal
SU	Soviet Union
TD	Chad
TG	Togo
US	United States of America

BNSDOCID: <WO___8001423A1_I_>

DATA PROCESSING APPARATUS HAVING OPCODE EXTENSION REGISTER

Background of the Invention

This invention relates to a central processor unit of a data processing system for executing a program of stored instructions through a sequence of instruction cycles, the central processing unit adapted for receiving an instruction during the instruction fetch phase of an instruction cycle and, responsive to the instruction for performing during the data execution phase of the instruction cycle a processor operation on stored data specified by the instruction, each stored instruction comprising an op-code portion and an address portion, the central processing unit comprising an instruction register for receiving the op-code portion of the instruction during the instruction fetch phase of the instruction cycle and storing the op-code throughout the entire instruction cycle and a controller for producing command signals corresponding to the processor operation during the data execution phase of the instruction cycle.

The op-code of an instruction is a group of binary digits (bits) that define a processor operation such as ADD, SUBTRACT, COMPLEMENT, etc. The set of processor operations formulated for a CPU depends on the processing it is intended to carry out. The total number of distinct operations which can be performed by the CPU determines its set of processor operations.

The number of bits which form the op-code (op-code field) is a function of the number of operations in the set. At least N bits are necessary to define 2^N or less distinct operations. The CPU designer assigns a different bit combination (i.e., op-code) to each operation. The controller section of the CPU detects the bit combination at the proper time in a sequence and produces proper command signals to required destinations in the CPU to execute the specified operation.

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

In addition to specifying a processor operation, an instruction will normally also carry other information such as the address(es) of memory locations where operand(s) to be used in the processor operations are stored. Ordinarily, the number of bits required for the operand address(es) (address field) occupy most of the bit positions available in the instruction leaving only a limited number of bits to be allocated for the op-code field. When a CPU designer finds the bits allocated to the op-code field insufficient for a given set of processor operations he has, heretofore, had the choice of either accepting a smaller set of processor operations or lengthening the instruction.

Long instructions are disadvantageous in small data processing systems where the memory capacity for storing instructions is limited. In addition, small systems have limited word sizes as well (as small as 4 bits in some systems where the CPU is in the form of a microprocessor), and a long instruction has the added disadvantage of requiring many memory references for retrieval and, thus, of slowing down CPU operation. However, from the standpoint of versatility, programming convenience, and operating efficiency, it is desirable to have a large set of processor operations. Therefore, a problem in designing a CPU for small data processing systems is that of being able to define a large set of processor operations while minimizing instruction length.

Prior art techniques for conserving the lengths of instructions have been directed towards reducing the length of the address field by using some form of abbreviated addressing of memory locations, such as the indirect addressing method or the relative addressing method of extending the address code. However, the problem remains on how to achieve additional reduction of the instruction length when an N-bit op-code field in the instruction defines more than 2^N processor operations.

BUREAU OMPI WIPO

BNSDOCID: <WO___8001423A1_I_>

The problem is solved in accordance with the central processing unit which further comprises an op-code extension register for storing a selected op-code extension word and the controller being responsive to the combined contents of the instruction register and the op-code extension register for producing command signals corresponding to the responsive combined contents, the contents of the op-code extension register being changed only when the central processing unit receives an instruction for transferring a newly selected op-code extension word to the op-code extension register.

Summary of the Invention

The present invention provides a CPU of a data processing system for executing a program of stored instructions through a sequence of instruction cycles, the CPU receiving an instruction during the instruction fetch phase of the instruction cycle and, responsive to the instruction, performing during the data execution phase of an instruction cycle an operation specified by the instruction on stored data specified by the instruction, each stored instruction comprising an op-code portion and an address portion, the CPU comprising an instruction register for receiving the op-code portion of the instruction during the instruction fetch phase and storing the op-code throughout the entire instruction cycle; and controller means for producing command signals corresponding to processor operations during the data execution phase, characterized in that the stored data comprises operands and op-code extension words, and there is included an op-code extension register for storing a selected op-code extension word, the controller means being responsive to the combined contents of the instruction register and the op-code extension register for producing command signals corresponding to processor operations required for the respective combined contents, the contents of the op-code extension register being changed when the CPU receives an instruction for transferring a newly

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

selected op-code extension word into the op-code extension register means.

Thus the contents of the op-code extension register, which need not change with each instruction cycle, is combined with the op-code stored in the instruction register, which changes with each new instruction, to form an effectively longer op-code.

Advantageous use of the op-code extension register concept is made by organizing the set of CPU operations into generic processor operations and optional CPU features which modify the generic operations. Therefore, the op-code field in each instruction defines the generic operations while the optional features are selected or not by the op-code extension word stored in the op-code extension register.

Accordingly, it is an object of the invention to provide a CPU architecture which permits a processor operation set which is larger than that which can be defined by the op-code field in an instruction.

It is another object of this invention to provide a CPU architecture which improves the versatility and performance of a small data processing system having a small word size and limited instruction storage capacity.

It is still a further object of this invention to provide a CPU architecture for reducing the number of memory references required to fetch an instruction in a data processing system having a small word size.

Another object of the invention is to provide a CPU architecture which allows the addition of software programmable optional features without requiring additional op-codes.

Yet another object of the instant invention is to provide a CPU architecture for low cost, high performance microprocessors.

The above and other objects of the invention are achieved in several illustrative embodiments described hereinafter. However, it should be appreciated that there

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

are possible useful embodiments which are designed to achieve less than all of the preceding objects while still remaining consistent with the principles of the invention. The novel features of the invention, both as to structure and method of operation, together with further objects and advantages thereof will be better understood from the following description considered in connection with the accompanying drawings. It is to be expressly understood, however, that the drawings are solely for the purpose of illustration and description and are not intended to define limits of the invention.

Brief Description of the Drawings

FIG. 1 is a block diagram illustrating a CPU known in the prior art.

FIG. 2 is a flow chart useful for illustrating the processor sequence during an instruction cycle.

FIG. 3 is a state diagram illustrating the sequence of steps followed by the CPU of FIG. 1 in executing an ALU operation.

FIG. 4 is a block diagram illustrating a CPU in which the operand width of the ALU is placed under program control in accordance with the instant invention.

FIG. 5 is a logic diagram of the Carry-In Administration Circuit, used in the CPU of FIG. 4.

FIG. 6 is a logic diagram of the Carry-Out Multiplexer Circuit used in the CPU of FIG. 4.

FIG. 7 is a logic diagram of the Carry-Out Register Circuit used in the CPU of FIG. 4.

FIG. 8 is a logic diagram of the Rotate/Shift Right Administration Circuit used in the CPU of FIG. 4.

FIG. 9 is a logic diagram of the ALU Function Inhibit Circuit used in the CPU of FIG. 4.

FIG. 10 is a block diagram illustrating a CPU in which the optional features of variable operand width, autoloading and autoincrementing of memory pointer registers are placed under program control in accordance with the instant invention.

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

FIG. 11 depicts the Op-code Extension Register used in the CPU of FIG. 10.

FIG. 12 is a state diagram illustrating the sequence of steps followed by the CPU of FIG. 10 in executing an ALU operation and the operation of the variable operand width and autoincrementing features.

FIG. 13 is a state diagram illustrating the sequence of steps followed by the CPU of FIG. 10 in address formation and the operation of the autoloading feature.

FIG. 14 is a block diagram illustrating a portion of a CPU in which the assignment of address register to store either memory addresses or peripheral device addresses is placed under program control in accordance with the principles of the instant invention.

FIG. 15 is a block diagram illustrating the Instruction Register, Op-Code Extension Register, and Controller of the CPU of FIG. 14 and a logic diagram of the circuit in the Controller which decodes the address register assignment field.

FIG. 16 is a block diagram depicting an interface arrangement of the CPU of FIG. 14 with a RAM and three peripheral devices.

Detailed Description

Referring now to FIG. 1 there is shown a block diagram representative of a simple CPU, 100, known in the prior art and of a type which is found in small data processing systems such as minicomputers and microprocessor based systems. Only those parts of the CPU which are essential to the explanation to follow have been included in FIG. 1. The configuration of FIG. 1 uses a single Instruction Register (IR), 101, which stores the current instruction being executed. A program of instructions to be executed by the CPU is stored in a Read-Only-Memory (ROM), 102, in the form binary words hereafter referred to as instruction words. In this example, each instruction word contains eight bits. The CPU retrieves an integral number of instruction words from the ROM to form a complete

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

instruction. Each instruction is composed of an op-code specifying an operation which the CPU is designed to perform and an address code specifying the memory location where an operand or operands to be used for the specified operation are stored. In the CPU of FIG. 1, the operands are stored in a Random Access Memory (RAM), 103, also in the form of binary words hereafter referred to as operand words. In this example, each operand word contains eight bits. An operand is formed with one operand word retrieved from the RAM. The IR holds eight bits of an instruction, six of which are reserved for the op-code, the remaining two being reserved for the part of the address code specifying the addressing mode. Four addressing modes are available in the CPU of FIG. 1, namely the direct mode, two indirect modes, and the immediate data mode. The length of an instruction will vary depending on the addressing mode used.

As an illustrative example, the 6-bit op-code is used to define 48 processor operations represented by the instruction set listed in Table 1.

Table I

CPU Instruction Set

	Instruction	Op-Code
1	ADD	000000
2	ADD WITH CARRY	000001
3	SUBTRACT	000010
4	SUBTRACT WITH CARRY	000011
5	AND	000100
6	OR	000101
7	EXCLUSIVE OR	000110
8	MOVE	000111
9	COMPARE AND BRANCH IF EQUAL	001000
10	COMPARE AND BRANCH IF NOT EQUAL	001001
11	COMPARE AND BRANCH IF LESS THAN	001010
12	COMPARE AND BRANCH IF GREATER THAN/EQUAL	001011
13	TEST AND BRANCH IF ZERO	001100

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

14	TEST AND BRANCH IF NOT ZERO	001101
15	DECREMENT, BRANCH IF NOT ZERO	001110
16	SET	001111
17	CLEAR	010000
18	INCREMENT	010001
19	DECREMENT	010010
20	COMPLEMENT	010011
21	ROTATE LEFT	010100
22	ROTATE LEFT WITH CARRY	010101
23	ROTATE RIGHT	010110
24	ROTATE RIGHT WITH CARRY	010111
25	SHIFT LEFT	011000
26	SHIFT LEFT WITH CARRY	011001
27	SHIFT RIGHT	011010
28	SHIFT RIGHT WITH CARRY	011011
29	SET BIT	011100
30	CLEAR BIT	011101
31	BRANCH ON BIT SET	011110
32	BRANCH ON BIT CLEAR	011111
33	EXCHANGE	100000
34	TEST WITH MASK AND BRANCH IF ZERO	100001
35	TEST WITH MASK AND BRANCH IF NOT ZERO	100010
36	BRANCH	100011
37	JUMP	100100
38	CALL	100101
39	RETURN	100110
40	RETURN ON INTERRUPT	100111
41	CLEAR FLAG REGISTER	101000
42	SET FLAG REGISTER	101001
43	PUSH	101010
44	POP	101011
45	EXCHANGE MEMORY POINTER	101100
46	LOAD STACK POINTER	101101
47	STORE STACK POINTER	101110
48	NO OPERATION	101111

The Program Counter (PC), 104, normally holds the

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

address of the next word of instruction or constant to be retrieved from the ROM. The PC is a 12-bit register which goes through a step by step counting sequence pointing to successive instruction words stored in the ROM, except in response to a program transfer instruction (e.g., instructions 9-15, 31, 32, 34-40, 43, and 44 of table 1) when a new address is loaded into PC. An 8-bit Data Bus, 105, is used for transferring instructions, addresses, and operands between various locations in the CPU. Addresses for accessing locations in the RAM, 103, area stored in the 12-bit Memory Address Latch (MAL), 106, which receives an address from one of a group of registers in the Memory Pointer (MP), 107. The contents of IR are detected by the Controller, 108, which produces command signals for processor operations and address manipulations specified by the detected contents. The command signals are transmitted to various destinations in the CPU via the control lines, 109.

The arithmetic and logical operations are performed in the Arithmetic Logic Unit (ALU), 110, which can receive either one or two operands at its A and B inputs. An Accumulator (ACC), 113, stores an operand for the ALU and also stores the result of the ALU operation. The ALU is provided with a Carry-In input to receive a carry signal and a Carry-Out output where the carry signal generated by the ALU is made available. The input carry signal is provided by the Carry-In Administration circuit, 111, which selects among the Plus 1 signal in the case of an INCREMENT operation, the Carry signal in the case of ROTATE LEFT WITH CARRY operation, or the most significant bit, AC7, of the Accumulator in the case of a ROTATE LEFT operation. The output carry signal is received by the Carry Register (CR), 112, which indicates an ALU function overflow or underflow. The CR is also loadable, resettable, and participates in the SHIFT and the ROTATE operations.

The ACC which has a master section, ACM, and a

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

slave section, ACS, is also used in this example as a latch for data to be read from or written into the RAM. A special Shift/Rotate Right Circuit (SRR), 114, provides for SHIFT AND ROTATE RIGHT operations involving ACC. The SHIFT AND ROTATE LEFT operation is accomplished by adding the operand to itself and, therefore, requires no special circuitry. A Rotate/Shift Right Administration Circuit (RSR), 115, is provided to select either the Carry Output signal or the least significant bit, AC0, of ACC to be loaded into the most significant bit position of ACC for the ROTATE RIGHT WITH CARRY and the ROTATE RIGHT operations respectively.

The execution of each instruction by a CPU occurs in a sequence of basic steps which form the instruction cycle. The sequence of steps is governed by timing signals generated by clock circuitry which is not shown in FIG. 1.

Referring now to FIG. 2 there is shown a flow chart of steps in the instruction cycle. The instruction cycle is divided into two phases: an instruction fetch phase and a data execution phase. Reference numerals 201 through 206 indicate the basic stages of each phase. In the instruction fetch phase, an instruction word whose address is in PC is transferred from the ROM to IR. The instruction word as previously mentioned has a 6-bit op-code field and a 2-bit address mode field. The first step in the data execution phase is the formation of the address or addresses of operands required for the operation. The operands are stored in RAM or in internal registers in the CPU. First, the op-code in IR is decoded by the controller to determine whether the operation is monadic or dyadic. A monadic operation is one which requires only one operand and, therefore, only one operand address is formed. The sole operand address formed for a monadic operation is called a destination address because the results of the operation are automatically stored in the memory location previously occupied by the sole operand. A dyadic operation is one which requires two operands and,

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

therefore, two operand addresses are formed. The address of the first operand is called the source address while that of the second operand is called the destination address because the results of the dyadic operation will be automatically stored in the location previously occupied by the second operand. In the CPU of FIG. 1, the destination address is generally always ACC.

The address mode bits in IR are also detected by the controller. If direct addressing is indicated, an address is transferred from a location in ROM into MAL. If, however, one of the two indirect addressing modes is indicated, the contents of a specified memory pointer register in MP is transferred to MAL. In the case of the immediate data addressing mode, the operand or operands themselves are carried by the address portion of the instruction and, therefore, no operand addresses are formed.

After the operand address is formed and transferred to MAL, the operand may be fetched from the location in RAM pointed to by MAL. The last step in the data execution phase is the performance of the operation specified by the op-code in IR which in this example is an ALU function. Completion of the ALU function terminates the instruction cycle.

Referring now to FIG. 3, a state diagram, 300, for the simplified CPU of FIG. 1 is shown. Reference numerals 301 through 310 indicate the detailed steps in an instruction cycle for executing ALU and data transfer operations. Non-data operations such as BRANCH, CALL, RETURN, etc. are classified as Miscellaneous Instructions and are not shown. The dyadic operation ADD between a first operand stored in a RAM location pointed to by a register in MP and a second operand already stored in ACC can be represented by the state sequence

S1 → S2 → S4 → S6 → S9 → S10.

Referring to the states represented by the blocks

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

designated by reference numerals 301 through 310 in FIG. 3, in state S1 the contents of the ROM location addressed by the contents of PC is transferred to IR. In state S2 the contents of IR are decoded by the Controller. Assuming in this example that the addressing mode is indirect, in state S4, 304, the contents of a specified register in MP are transferred into MAL. In state S6, 306, the operand stored in the RAM location specified by the address in the MAL is transferred to the ALU, and a particular ALU operation specified by the op-code (i.e., ADD) is performed on the above operand and the contents of ACC, with the result transferred to the master section (ACM) of ACC. In state S9, 309, the contents of ACM may, as required by the instruction, be transferred either to the slave (ACS) of ACC or to a RAM location pointed to by MAL. For the ADD operation the result (the sum) remains in ACC. In state S10, 310, PC is incremented before the beginning of a new instruction cycle.

An example of a Monadic operation, SET ACCUMULATOR, can be represented by the state sequence

S1 → S2 → S7 → S9 → S10.

The steps represented by the states S1 and S2 are the same as described above for the dyadic operation. In state S7, 307, the ALU generates an 8-bit word having all "1s" and transfers it into ACM. The steps represented by states S9 and S10 are the same as described above for the dyadic operation.

An example of a data transfer operation, MOVE, of a constant stored in the ROM to a register in MP is represented by the state sequence

S1 → S2 → S3 → S5 → S10.

The steps represented by states S1 and S2 have already been explained in connection with the dyadic operation. In

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

state S3, 303, PC is incremented to point to the ROM location where the constant is stored. In state S5, 305, the constant in the ROM location pointed to by PC is transferred to MP. The step represented by state S10 has already been explained in connection with the dyadic operation.

The CPU configuration of FIG. 1 can be modified to include processor operations in addition to those listed in Table 1. For example, as will be described in detail, the ALU section can be altered to accommodate 4-bit operands as well as the normal 8-bit operands with the selection of the operand width under program control. Such a modification would affect the arithmetic, logical and data movement operations represented by instructions 1 through 35 in Table 1. The modification would effectively add thirty-five new operations for 4-bit operand width to those listed in Table 1, raising the total number of operations in the set to eighty-three. Inasmuch as the 6-bit op-code field in the CPU of FIG. 1 permits only sixty-four distinct codes, the op-code field must be extended to accommodate the additional operations. Owing to the fact that the word size of the CPU is 8-bits, to avoid interfacing difficulties, it would be necessary to extend the IR by multiples of 8-bits. Therefore, if each instruction were to carry information concerning the operand width, the instruction length would have to increase by eight bits. This would cause a considerable increase in ROM space required for storing a program and would also require one more ROM reference to be added to each instruction cycle. The additional memory overhead increases system cost while the additional ROM references slow down system operating speed.

An alternative means for program selection of the operand width is provided by the principles of the instant invention. It may be recognized that in most programs, when a particular operand width is selected, many operations may be executed before it becomes necessary to

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

change the operand width. Thus the portion of the op-code field which specifies the operand width remains constant over many instruction cycles while the portion of the op-code field which specifies the operations listed in Table 1 (generic operations) ordinarily changes with each new instruction cycle. Therefore, it would be unnecessary to carry in each instruction the infrequently changing portion of the op-code, which may instead be stored in a special hardware register. For present purposes this special hardware register will be called the op-code extension register (OER). Use of the OER for storing the portion of the op-code which specifies the operand width means that the op-code field in each instruction need only specify the generic operations, for which six bits are sufficient. Therefore, by using the OER concept, it becomes unnecessary to increase the instruction length.

In accordance with the above disclosed concept, the OER may also specify other optional features in addition to variable operand width. Some of these other features such as autoloading and autoincrementing of memory pointer registers, and the assignment of address registers, will also be described in detail in this specification.

When there are several optional features to be specified by the OER, there must be a sufficient number of bit positions in the OER to specify all distinct combinations of optional features. A group of bits to be stored by the OER is hereafter referred to as an op-code extension word. The contents of OER are changed only when necessary by means of special instructions (added to the list of Table 1) for transferring to OER a new op-code extension word from memory, and for changing a particular bit in OER. Each op-code extension word corresponds to a distinct combination of special features to be used in the program. Thus, by using the OER concept for program selection of optional features, ROM space is conserved and additional ROM references in each instruction cycle are avoided.

BUREAU OMPI WIPO

BNSDOCID: <WO___8001423A1_I_>

Variable Operand Width

Referring now to FIG. 4, there is shown a block diagram representative of a CPU, 400, basically similar to that of FIG. 1 but with modification to permit changing of the operand width from 8 bits to 4 bits. These modifications include dividing the ALU, the Accumulator, and the Shift and Rotate Right Circuit into two independent 4-bit sections which will be referred to as the "higher" and "lower" sections. In FIG. 4, the "higher" sections of the ALU (ALUH), Accumulator (ACCH), and the Shift and Rotate Right Circuit (SRRH), are indicated by reference numerals 410, 416, and 418 respectively. The "lower" sections of the ALU (ALUL), the Accumulator (ACCL), and the Shift Right and Rotate (SRRL) are indicated by reference numerals 411, 417, and 419, respectively. In each case the "higher" and "lower" sections are connected such that when the operand width is 8-bits, the "higher" section operates on the most sigificant 4-bits of the operand word while the "lower" section operates on the least significant 4-bits of the 8-bit operand word. When the operand width is 4-bits, the "higher" sections of each divided part are rendered inoperative as if the current instruction were NO OPERATION, while the "lower" sections operate on the least significant 4 bits of an operand word.

The Carry-In Administration Circuit (CIA), 412, normally provides a "0" to the Carry-In input of ALUL, 411. When the current instruction is INCREMENT, or DECREMENT, the CIA output is a "1". The CIA output is the state of the Carry Register, 413, when the current instruction is ADD WITH CARRY, SUBTRACT, or ROTATE LEFT WITH CARRY. For the ROTATE LEFT instruction, the CIA output is the most significant bit (AC7) of ACCH in the case of an 8-bit operand and is the most significant bit (AC3) of ACCL in the case of a 4-bit operand. An example of a logical implementation of the CIA is shown in FIG. 5.

The Carry-Out Multiplexer Circuit, (COMX), 414, provides the Carry-Bit input to the Carry Register, 413.

BUREAU

BNSDOCID: <WO__8001423A1_I_>

The COMX output is either the Carry-Out output of ALUH in the case of an 8-bit operand or the Carry-Out output of ALUL in the case of a 4-bit operand. An example of a logical implementation of the Carry-Out Multiplexer is shown in FIG. 6.

The Carry Register (CR), 413, is a flip-flop which can be set from several sources depending upon the current instruction. The operative inputs of CR for various instructions are listed below:

Instruction	Operative CR Input
(ALU Operation) WITH CARRY	CB from COMX
LOAD CR	Data Bus
CLEAR CR	"0"
SHIFT RIGHT WITH CARRY	"0"
ROTATE RIGHT WITH CARRY	AC0.

An example of a logic implementation of the carry register is shown in FIG. 7.

The Rotate/Shift Right Administration Circuit (RSR), 420, provides the most significant bit for ACCH and/or ACCL in SHIFT AND ROTATE RIGHT operations. In the case of 8-bit operand width, RSR provides the state of the least significant bit (AC0) of ACCL to the most significant bit position (AC7) of ACCH via the OT8 output. In addition RSR provides the state of the least significant bit (AC4) of ACCH to the most significant bit position (AC3) of ACCL via the OT4 output. In the case of 4-bit operand width, RSR provides the state of the least significant bit (AC0) of ACCL to the most significant bit position (AC3) of ACCL via the OT4 output. The inputs and operative outputs of RSR for various instructions are listed below:

Instruction	Inputs	Operative Outputs
Shift Right (8-bit)	0,AC4	OT8,OT4
Shift Right (4-bit)	0	OT4
Rotate Right (8-bit)	AC0,AC4	OT8,OT4
Rotate Right (4-bit)	AC0	OT4

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

Right Shift/Rotate with carry (8-bit)	carry,AC4	OT8,OT4
Right Shift/Rotate with carry (4-bit)	carry	OT4.

A logical implementation of the Rotate/Shift Right Administrative Circuit is shown in FIG. 8.

The ALU Function Inhibit Circuit (ALUFI), 415, is used to disable ALUH when the operand width is 4-bits. When the operand width is 8-bits, the ALU function control lines, 422, from the controller, 408, govern the selection of ALU functions for both ALUH and ALUL. However, when the operand width is 4-bits, ALUFI applies a control signal to ALUH which corresponds to the NO OPERATION instruction, and ALUH allows operands to pass through unchanged. An example of a logical implementation of the ALU Function Inhibit Circuit is shown in FIG. 9.

When 4-bit operand width is selected for the CPU of FIG. 4, only the least significant 4-bits of the operand word are operated on while the most significant 4-bits are unused. In order to make use of the most significant 4-bits of the operand word and thereby make full use of the RAM storage space which is organized to store 8-bit words it is necessary to provide means to interchange the most significant and least significant 4-bits of the operand word. Such an interchange can be achieved in several ways.

One way to achieve an effective interchange of the most and least significant 4-bits in an operand word is to interchange, under the control of a bit position in the OER, the "higher" and "lower" sections of the ALU. For example, if a designated bit position of the OER for controlling the ALU interchange contains a "0" during 4-bit operation, ALUFI would inhibit ALUH while ALUL is allowed to operate on the least significant 4-bits of the operand word. However, if the designated bit position contains a "1" during 4-bit operation, ALUFI would inhibit ALUL while ALUH is allowed to operate on only the most sigificant 4-

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

bits of the operand word.

The most and least significant 4-bits of an operand word can be interchanged directly by introducing a ROTATE 4 instruction to the list in Table 1. The CPU of FIG. 4 upon receiving such an instruction transfers an operand word specified by the instruction from RAM to the divided accumulator where initially the most significant 4-bits of the operand word reside in ACCH and the least significant 4-bits reside in ACCL. The operand word is then rotated in the divided accumulator until the previous most significant 4-bits reside in the ACCL and the previous least significant 4-bits reside in ACCH. The "rotated" operand word in the divided accumulator can either be immediately used in an ALU operation or be transferred back to RAM.

Programmable selection of the operand width is by means of the op-code extension register (OER), 421, which in this example need only contain one bit position. When the binary state of that bit is a "1", the operand width for ALU and data movement operations is 4-bits, otherwise the operand width is 8-bits. A LOAD OER instruction is added to the list in Table 1, and the ROM contains two constants, a "1", and a "0", as op-code extension words. Although only one bit is needed to specify the operand width, OER may contain additional bit positions for selecting other optional features as discussed above.

An alternative CPU configuration for implementing variable operand width as well as other optional features is shown in FIG. 10. Although the CPU architecture of FIG. 10 is basically different from that of FIG. 4, the same op-code extension register concept is used to select the optional features. Referring now to FIG. 10, the Instruction Register, 1001, the ROM, 1002, the RAM, 1003, the Data Bus, 1004, and the Controller, 1005, all serve the same function as their corresponding parts in the CPU of FIG. 4 except the word size of the ROM and RAM, and the widths of the Data Bus and the Instruction Register are all

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

4-bits instead of 8-bits. The Address Latch (AL), 1006, is a 12-bit master-slave latch for storing the current address of an instruction or of an operand. The 12-bit address arithmetic unit (AAU), 1007, increments or decrements the current address in AL. The Internal Register Memory (IRM), 1008, is a group of registers which include a Program Counter (PC), 1009, two Memory Pointer Register B0, 1010, and B1, 10011, and a Stack Pointer, 1012. The Temporary Register Memory (TRM), 1013, contains two temporary registers T0, 1014, and T1, 1015, used for intermediate address calculations during the address formation stage. The four address modes available in the CPU of FIG. 4 are also available in the CPU of FIG. 10.

The Data-Address Bus Multiplexer (DAMUX), 1016, controls the transfer of data in 4-bit words from ROM and RAM to the 12-bit registers in RIM and TRM via the 4-bit Data Bus. The DAMUX multiplexes the 4-bits from the Data Bus onto one of three 4-bit sections (designated higher, middle and lower sections) of the Address Arithmetic Bus, 1026. The three sections of the Address Arithmetic Bus are coupled respectively to the higher, middle, and lower 4-bit sections of the 12-bit registers in IRM and TRM. A 12-bit address bus, 1027 transfers addresses from the AL to the AAU, RAM, and ROM.

The 4-bit Arithmetic and Logic Unit (ALU) is used to perform all arithmetic and logical functions. The operands for the ALU are stored in 4-bit temporary data registers TA, 1018, and TB, 1019.

Unlike the CPU of FIG. 4, the CPU of FIG. 10 does not have an addressable accumulator. Instead, the RAM locations pointed to by the Memory Pointer Registers, 1010 and 1011, are allowed to be used as accumulators. This permits the CPU to have as many accumulators as RAM space permits.

The basic operand width of the ALU in the CPU of FIG. 10 is 4-bits; however, its configuration is such that ALU operations can be performed on operands whose widths

BUREAU OMPI

BNSDOCID: <WO__8001423A1_I_>

are integral multiples of the basic width (e.g., 8, 12, and 16-bits). For operands having widths which are multiples of the basic width, the ALU operates on 4-bit segments of the operand beginning with the least significant segment and repeating the ALU operation on the other segments according to their order of significance. A 4-bit segment is commonly referred to as a nibble. The number of repetitions of an ALU operation required for a given operand width is equal to the number of nibbles in the operand.

In the example of FIG. 10, OER has six bit positions which are assigned according to FIG. 11. Referring to FIG. 11 a 2-bit field comprising bits b0, 1101, and b1, 1102, specify the four optional operand widths of 4-bits, 8-bits, 12-bits, and 16-bits. The other 4-bit positions of OER are used to specify other special features to be described later in this specification. The coding of bits b0 and b1 and the number of repetitions needed to complete an ALU operation for each state are listed below:

b1, b0	operand width	ALU repetitions
00	16-bits	4
01	4-bits	1
10	8-bits	2
11	12-bits	3

Referring again to FIG. 10, the number of repetitions of an ALU operation specified by the contents of OER is controlled by the 2-bit Counter, 1021, and the Comparator, 1022, which compares the state of the counter with that of b0-b1. When the state of the counter and that of b0-b1 match, repetition of the ALU operation ceases.

The details of an ALU operation in the CPU of FIG. 10 can be better understood when considered with the State Diagram, 1200, shown in FIG. 12. Referring to FIG. 12, the Instruction Fetch States, 1201, and the Address Formation States, 1202, have already been discussed

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

in connection with FIG. 2. After address formation, the source address for a dyadic operation is stored in temporary register T0. With the operation specified by the current instruction as either monadic or dyadic, the destination address is in temporary register T1. It is to be noted that in the CPU of FIG. 10 a destination address is always required for a dyadic operation, unlike the CPU of FIG. 4 where the accumulator is always the implied destination for dyadic operations.

In state ALU1, 1203, the contents of T0 (source address) is transferred to the master (ALM) of the Address Latch (1023 in FIG. 10). If the current instruction specifies a monadic operation, the next state is ALU5, 1207; otherwise the next state is ALU2, 1204. During the transition from ALU1 to ALU2, the contents of ALM are transferred to the slave (ALS) of the Address Latch (1024 in FIG. 10), and in state ALU2 the operand stored in the RAM location pointed to by ALS is transferred to temporary data register TB. In state ALU3, 1205, the address in ALS is incremented and transferred to both T0 and ALM, and if another optional feature, autoincrementing, yet to be described, is enabled, the incremented address is also transferred to pointer register B0. If the current instruction specifies immediate data addressing for the second operand of the dyadic operation (i.e., the second operand itself is carried in the address field of the current instruction) the next state is ALU4, 1206. In ALU4 the contents of ALS are transferred to PC, restoring the ROM address for the next data constant or instruction. For the other addressing modes, the next state is ALU5, 1207, where the destination address stored in T1 is transferred to ALM and the 2-bit Counter is incremented, the Counter having been cleared at the beginning of the instruction cycle. During the transition from state ALU5 to state ALU6 the contents of ALM are transferred to ALS. In state ALU6, 1208, the contents of the RAM location pointed to by ALS are transferred to temporary data register TA. The next

BUREAU OMPI

BNSDOCID: <WO__8001423A1_I_>

state is ALU7, 1209, where ALS is incremented and its contents transferred to T1 and to B1 if the autoincrementing feature is enabled. The ALU operation specified by the current instruction is performed on two operands provided by TA and TP in the case of a dyadic operation or on a single operand provided by TA in the case of a monadic operation. The next state is ALU8, 1210, where the result of the ALU operation (ALU OUT) is stored in the RAM location pointed to by ALS. The state of the Counter is then compared with the state of b0=b1 in OER. If the Comparator output is true, the ALU operation is completed; but if the Comparator output is false, the CPU returns to state ALU1 and the ALU operation is repeated.

Auto Loading of Pointer Registers

Another optional feature of the CPU in FIG. 10 which can be selected under program control using the OER is autoloading of the pointer registers B0 and B1. This feature is only available for the direct addressing mode where the full operand address is carried in the address code of the instruction. When autoloading is enabled, the operand addresses supplied by the current instruction are automatically stored in specified pointer registers at the completion of the instruction cycle. Autoloading of B0 and B1 is controlled by the state of bit positions AL0 (1105 of FIG. 11) and AL1 (1106 of FIG. 11) in OER respectively, a "1" state enabling the autoloading feature.

Autoloading is performed during the address formation stage of the data execution phase. A state diagram of the address formation stage for the CPU of FIG. 10 is shown in FIG. 13. An example of address formation is now explained with reference to FIG. 13.

During the instruction fetch phase represented by the block with reference numeral 1301, an instruction word containing an op-code and an address mode code is transferred from ROM to IR. The CPU then goes to the first address formation state AF1, 1302. The path followed by the CPU through the address formation steps depend on the

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

addressing mode specified by the address mode code in IR and on whether the operation specified by the op-code in IR is monadic or dyadic. A D-type flip-flop (DFF), 1025, in the CPU of FIG. 10 keeps track of whether the address being formed for a dyadic operation is a source or destination address. Prior to the formation of the source address of a dyadic operation, DFF is clear (DFF=0), but prior to the formation of the sole operand address of a monadic operation or the destination address of a dyadic operation, DFF is set (DFF=1). If the direct addressing mode is specified for the source address, the most significant (upper) 4-bits of T0, are set to all "1s", and, similarly, if the direct addressing mode is specified for the destination address, the most significant 4-bits of T1 are set to all "1s".

The CPU then proceeds to state AF2, 1303. If the specified addressing mode is for indirect addressing, the contents of B0 are transferred to T1 if DFF is set, or the contents of B0 are transferred to T0 and the contents of B1 are transferred to T1 if DFF is clear. The CPU then proceeds directly to state AF6, 1307. If the specified addressing mode is for immediate data, the state of PC is transferred to T0 and the CPU proceeds directly to state AF6.

In state AF2, for the case of direct addressing, the contents of PC are transferred to ALM, the contents of PC being the address of the ROM location containing the first nibble of the operand address. During the transition from state AF2 to state AF3, 1304, the contents of ALM are transferred to ALS. In state AF3, ALS is incremented and its contents transferred to PC. If another nibble from ROM is required to complete the formation of the address the incremented address in ALS is also loaded into ALM. The contents of the ROM Location pointed to by ALS are then transferred to the lower four bits of T0 (T0L) if DFF is set, or to the lower four bits of T1 (T1L) if DFF is clear. During the transition from state AF3 to state AF4, 1305,

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

the contents of ALM are transferred to ALS. In state AF4, ALS is again incremented and its contents transferred to PC. The contents of the ROM Location pointed to by ALS are transferred to the middle four bits of T0 (T0M) if DFF is set, or to the middle four bits of T1 (T1M) if DFF is clear. It is to be noted that the upper four bits of an operand address in the direct addressing mode is "1111".

If the autoloading feature is not enabled for either B0 or B1, then the CPU proceeds from state AF4 directly to state AF6. Otherwise, the next state is AF5, 1306. In state AF5 the contents of T0 is loaded into B0 if DFF is set and autoloading is enabled for B0 (i.e., AL0=1), or the contents of T1 is loaded into B1 if DFF is clear and autoloading is enabled for B1 (i.e., AL1=1).

In state AF6, the completed operand address in T0 is transferred to ALM. If DFF is set in state AF6, address formation is completed and the CPU proceeds to the ALU function represented by the block with reference numeral 1309. If, however, DFF is clear indicating that address formation for a dyadic operation is incomplete, the CPU proceeds first to state AF7 where DFF is set and then to state AF1 and repetition of the address formation steps for the destination address.

Autoincrementing of Pointer Registers

Another optional feature of the CPU of FIG. 10 which can be enabled or disabled by means of the OER is the autoincrementing of the Pointer Registers B0 and B1. When this feature is enabled for a specified Memory Pointer Register, the address stored in that register will be automatically advanced (incremented) at the end of each instruction cycle to point to the memory location of the first nibble of the next operand. Autoincrementing of B0 and B1 is controlled by the state of the OER bit positions AI0 (1103 of FIG. 11) and AI1 (1104 of FIG. 11), respectively. A "1" state enables the autoincrementing feature.

When enabled, autoincrementing of B0 and B1 take

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

place during the ALU function stage of the data execution phase as represented by the state diagram of FIG. 12. Referring now to FIG. 12, as explained above in connection with the variable operand width feature, when the CPU is in state ALU3, 1205, a source address stored in ALs pointing to a nibble of a first operand (assuming a dyadic operation) is incremented to point to the next nibble of the same operand, if the ALU operation is to be repeated on the next nibble. If the ALU Operation has been completed on all nibbles of the same operand, ALS points to the first nibble of a new first operand. The incremented address is then transferred to T0 and ALM. If autoincrementing of B0 has been enabled, (i.e., AI0=1), the incremented address is also transferred from ALS to B0. In state ALU7, 1209, a destination address contained in ALS pointing to a nibble of the second operand of the dyadic operation, is incremented and transferred to T1. If autoincrementing of B1 has been enabled (i.e., AI1=1), the incremented address is also transferred from ALS to B1. The ALU operation is then performed on the corresponding nibbles of the first and second operands residing respectively in TA and TB. The result of the ALU operation is stored in the memory location previously occupied by the corresponding nibble of the second operand. The ALU operation is repeated, if required, on additional nibbles of the first and second operands. It is to be noted that with autoincrementing enabled for a particular pointer register its contents are incremented with each repetition of the ALU operation such that at the end of the instruction cycle the pointer register is always pointing to the first nibble of the next operand.

Assignment of Address Registers

Yet another optional feature which may be placed under program control by means of OER is the assignment of address registers. In some CPU configurations, the RAM and peripheral devices (e.g., teletype keyboard, card reader, printer, magnetic tape drive, etc.) coupled to the CPU

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

share not only the same address field in the instruction but in addition they share the same addresses. That is, a given address may refer to either a RAM location or a peripheral device. Identification of an address as a RAM location or a peripheral device can be achieved by defining separate and distinct RAM reference and peripheral reference instructions (e.g., READ RAM, READ PERIPHERAL, etc.). However, in order to conserve op-codes, the same instruction can be used to reference both RAM and peripheral devices. Accordingly, the instruction would specify whether the address is for a RAM location or for a peripheral device by referencing either an address register specifically reserved to store a memory address or one specifically reserved to store a peripheral device address.

In prior art CPUs using the reserved address register architecture, the numbers of memory address registers and peripheral address registers are fixed by the CPU architecture. That is, the CPU designer decides on the number of each type of address register based on the type of applications for which the CPU was intended. For example, if the CPU were to be used in memory intensive application, it would have more memory address registers and fewer peripheral address registers. However, a fixed assignment of address registers limits the versatility of a CPU.

Optimal versatility can be achieved in both memory intensive and peripheral intensive applications by making the assignment of the address register variable under program control. Such an optional feature, permitting the assignment of address registers to be changed repeatedly during program execution, is particularly advantageous for programs which are memory intensive in some parts and peripheral intensive in others.

Referring now to FIG. 14, there is shown a block diagram representative of a portion of a CPU, 1400, which permits programmable assignment of several hardware address registers in accordance with the principles of this

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

invention. A group of eight 12-bit address registers, B0-B7 designated by reference numerals 1401 through 1408, respectively, can each be assigned to store either a memory location address or a peripheral device address. The functions of the Data Bus, 1409, Program Counter, 1410, Instruction Register, 1411, serve essentially the functions as explained in connection with the CPU of FIG. 4. The Op-code Extension Register (OER), 1412, stores an op-code extension word which specifies the selection of optional CPU features such as the ones discussed above including the assignment of the address registers to store RAM addresses and others to store peripheral device addresses is specified by the states of three bit positions in OER. The Controller, 1413, simultaneously decodes the contents of both IR and OER, effectively appending one to the other, and producing Command signals corresponding to the processor operation required by the combined contents of IR and OER. The Address Multiplexer, 1418, couples the address register referenced by the instruction to the 12-bit Address Out Bus, 1417.

When an instruction references a particular address register, the Controller enables the appropriate address register select line, 1415, and Address MUX Control Line making the address stored in that register available on the Address Out Bus. At the same time, the Controller decodes the contents of OER to determine whether the referenced address register holds a RAM address or a peripheral device address. If it holds a RAM address, the RAM Select Control Line, 1419, is enabled; otherwise the Peripheral Select Control Line, 1420, is enabled. The Read/Write Control Line, 1421, provides a signal to RAM indicating whether a memory reference requires data to be read from or written into the RAM.

Referring now to FIG. 15, there is shown a logic diagram of the portion of the Controller, 1500, which decodes the address register assignment field, AR0-AR2, of OER. If the instruction calls for the referencing of one

BUREAU OMPI WIPO INTERNATIONAL

of the address registers, B0-B7, the Instruction Decoder section, 1503, will cause the I/O Instruction Enable Line, 1504, and the Address Register Select Line, 1505, to go into the "1" state. In the example of FIG. 15, the eight address registers, B0-B7, are assigned two at a time by the three bits (AR0-AR2) in the address register assignment field of OER. Registers B0 and B1 are always reserved for RAM addresses, while registers B2 and B3 are assigned for RAM address if bit AR2, 1504, is a "0" and are assigned for peripheral device addresses if bit AR2 is a "1". In a similar manner the state of bit AR1 determines the assignment of registers B4 and B5 and the state of bit AR0 for registers B6 and B7. The circuit composed of logic gates G1-G10 (reference numerals 1506-1515) detects the states of AR0-AR2 and those of the address register select lines, 1505, to provide the proper states on the RAM Select Line or the Peripheral Select Line depending on the assignment of the selected address register.

In FIG. 16, there is shown an example of an input/output (I/O) interface arrangement of the CPU of FIG. 14 with a RAM and three peripheral devices. In this example, it has been assumed that the CPU has only one I/O Port, the RAM, 1602, has its own internal address decoder, and the three peripheral devices, 1603-1605, are all input devices which provide data to the CPU. When the CPU executes an input data instruction referencing one of the address registers, the address stored in the referenced address register is made available via the Address Out output of the CPU to the Peripheral Address Decoder (PAD) 1606, and to the Address Input of the RAM. Assuming in this example that the address provided at Address Out refers to both Peripheral Device 2, 1604, and a location in the RAM, in order to choose between the two the Controller in the CPU detects the address register assignment field (AR0-AR2) of OER to determine whether the selected address register has been assigned to RAM or to peripherals. The contents of OER have been loaded during a previous

BUREAU OMPI WIPO INTERNATION

BNSDOCID: <WO___8001423A1_I_>

instruction.

If the selected address register has been assigned to peripherals, the RAM is disabled via the RAM Select output while the PAD is enabled via the Peripheral Select output. The PAD activates Device 2 and causes the Peripheral Device Multiplexer, 1607, to couple Device 2 to the I/O port of the CPU. If however, the selected address register were assigned to RAM, the PAD would be disabled and the RAM enabled. A read signal provided by the Read/Write output would cause the RAM to transfer the contents of the addressed memory location to the CPU via the I/O Port.

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

Claims

1. A central processing unit of a data processing system for executing a program of stored instructions through a sequence of instruction cycles, the central processing unit adapted for receiving an instruction during the instruction fetch phase of an instruction cycle and, responsive to the instruction for performing during the data execution phase of the instruction cycle a processor operation on stored data specified by the instruction, each stored instruction comprising an op-code portion and an address portion,

the central processing unit comprising:

an instruction register for receiving the op-code portion of the instruction during the instruction fetch phase of the instruction cycle and storing the op-code throughout the entire instruction cycle; and

a controller for producing command signals corresponding to the processor operation during the data execution phase of the instruction cycle,

CHARACTERIZED IN THAT

the central processing unit (100) further comprises:

an op-code extension register (421) for storing a selected op-code extension word;

the controller (408) being responsive to the combined contents of the instruction register (401) and the op-code extension register (421) for producing command signals corresponding to the respective combined contents, the contents of the op-code extension register being changed only when the central processing unit (100) receives an instruction for transferring a newly selected op-code extension word to the op-code extension register (421).

2. A central processing unit in accordance with claim 1 further comprising:

a memory for storing the instructions for the data, the memory comprises multiple memory locations, each

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

having a distinct location memory location address,
the central processing unit comprises:
an address bus coupled to an address decoder associated with the memory and responsive to an address carried by the address bus for selecting the memory location corresponding to the address
CHARACTERIZED IN THAT
the central processing unit comprises:
an arithmetic logic unit (411) responsive to select command signals corresponding to the contents of the op-code extension word register and the instruction register from the controller for receiving one or more operands from the memory and for performing on the operand or operands an arithmetical or logical operation specified by the combined contents of the instruction register and the op-code extension register, the arithmetic logic unit (411) being adaptable to accommodate operands having one or more N-bit segments, the number of N-bit segments in an operand being specified by the op-code extension word stored in the op-code extension register.

3. A central processing unit as recited in claim 2
CHARACTERIZED IN THAT
the central processing unit receives 2N-bit operand words from the memory and the arithmetic logic unit having an upper and a lower portion for operating on the most significant N-bit segment and the least significant N-bit segment respectively of a 2N-bit operand word retrieved from the memory; when a designated bit position in the op-code extension register contains a first binary state, both portions of the arithmetic logic unit are enabled to operate on the entire 2N-bit operand word, and when the designated bit position contains the other binary state, the upper portion of the arithmetic logic unit is disabled while the lower portion is enabled to operate on only the least significant N-bit segment of the operand word leaving the

BUREAU OMPI WIPO INTERNATIONAL

most significant N-bit segments unchanged.

4. A central processing unit in accordance with claim 3

CHARACTERIZED IN THAT

circuitry for interchanging one for the other the most significant and least significant N-bit segments of a 2N-bit operand word.

5. A central processing unit in accordance with claim 2 comprising:

(1) at least one bidirectional input/output port being adapted to be coupled to a plurality of peripheral devices, each having a distinct peripheral device address and to a portion of the memory, some of the memory locations in the portion of the memory having addresses which are in common with peripheral device addresses;

(2) an address output coupled to the address bus and adapted to be coupled to peripheral device address decoder associated with the peripheral devices, the peripheral device address decoder being responsive to an address provided at the address output for selecting a peripheral device to be coupled to the input/output port, and to a portion of the memory address decoder responsive to the address at the address output for selecting a memory location in the portion of the memory to be accessed by the central processing unit through the input/output port;

(3) a first output adapted to be coupled to the peripheral address decoder for enabling the peripheral address decoder when the first output is in a first binary state and for disabling the peripheral address decoder when the first output is in a second binary state;

(4) a second output adapted to be coupled to the portion of the memory address decoder for enabling the portion of the memory address deocder when the second output is in a first binary state and for disabling the

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

portion of the memory address decoder when the second output is in a second binary state;

(5) a plurality of address registers for storing addresses, at least one of the address registers being assignable to store either a memory address corresponding to a memory location in the portion of the memory or a peripheral device address,

CHARACTERIZED IN THAT

the op-code extension register specifies the assignment of each assignable address register when an instruction being executed by the central processing unit specifies the transfer of the contents of a particular address register to the address output;

the controller responsive to the contents of the op-code extension register providing a first binary state at the first output and a second binary state at the second output if the particular address register is assigned to store a peripheral device address; and

the controller providing a second binary state at the first output and a first binary state at the second output if the particular address register is assigned to store a memory address corresponding to a memory location in the portion of the memory.

BUREAU OMPI

BNSDOCID: <WO__8001423A1_I_>

AMENDED CLAIMS
(received by the International Bureau on 18 June 1980 (18.06.80))

1. A central processing unit of a data processing system designed for executing a program of stored instructions through a sequence of instruction cycles, the central processing unit being adapted to be coupled to memory means (1002, 1003) having a multiplicity of addressable locations for storing instructions and data, each instruction having an op-code portion and an address portion, the data including operands, the central processing unit comprising:

instruction register means (1001, 1015, 1014) for storing an instruction;

instruction retrieval means (1008, 1006, 1007, 1026) for transferring a present instruction from the memory means to the instruction register means in each instruction cycle;

a controller responsive to the op-code portion of the present instruction for generating control signals corresponding to processor operations specified by the op-code portion;

data retrieval means responsive to particular control signals and to the address portion of the present instruction for transferring data specified by the address portion from the memory means to the central processing unit;

CHARACTERIZED IN THAT

the data includes operands and op-code extension words;

the central processing unit further comprises:

an op-code extension register (1020) for receiving and storing an op-code extension word;

the controller (1005, 1021, 1022, 1025) being responsive to the combination of the op-code portion of the present instruction and the contents of the op-code extension register for generating the control signals; and

the data retrieval means (1008, 1006, 1007, 1026) for transferring an op-code extension word specified by the address portion of the present instruction from the memory means to the op-code extension register when the present instruction is a special instruction for transferring a newly selected op-code extension word to the op-code extension register.

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

2. A central processing unit in accordance with claim 1

CHARACTERIZED IN THAT

the central processing unit comprises:

an arithmetic logic unit (411) responsive to select command signals corresponding to the combination of the contents of the op-code extension word register and the op-code portion of the present instruction from the controller for receiving one or more operands from the memory and for performing on the operand or operands an arithmetical or logical operation specified by the combination, the arithmetic logic unit (411) being adaptable to accommodate operands having one or more N-bit segments, the number of N-bit segments in an operand being specified by the op-code extension word stored in the op-code extension register.

3. A central processing unit in accordance with claim 2

CHARACTERIZED IN THAT

the operands for the arithmetic and logic operations each have 2 N-bit segments and the arithmetic logic unit have an upper and a lower portion for operating on the most significant N-bit segment and the least significant N-bit segment of each operand, respectively, both the upper and lower portions being enabled to operate on both segments of an operand at one time when a first designated bit position in the op-code extension register contains a first binary state, the upper portion being disabled and the lower portion being enabled to operate on only the least significant segment of an operand leaving the most significant segment unchanged when the first designated bit position contains a second binary state.

4. A central processing unit in accordance with claim 3

CHARACTERIZED IN THAT

circuitry for interchanging one for the other the most significant and least significant N-bit segments of each operand.

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1_I_>

5. A central processing unit in accordance with claim 2 comprising:

(1) at least one bidirectional input/output port being adapted to be coupled to a plurality of peripheral devices, each having a distinct peripheral device address and to a portion of the memory, some of the memory locations in the portion of the memory having addresses which are in common with peripheral device addresses;

(2) an address output coupled to the address bus and adapted to be coupled to a peripheral device address decoder associated with the peripheral devices, the peripheral device address decoder being responsive to an address provided at the address output for selecting a peripheral device to be coupled to the input/output port, and to a memory address decoder responsive to the address at the address output for selecting a memory location in the portion of the memory to be accessed by the central processing unit through the input/output port;

(3) a first output adapted to be coupled to the peripheral address decoder for enabling the peripheral address decoder when the first output is in a first binary state and for disabling the peripheral address decoder when the first output is in a second binary state;

(4) a second output adapted to be coupled to the portion of the memory address decoder for enabling the portion of the memory address decoder when the second output is in a first binary state and for disabling the portion of the memory address decoder when the second output is in a second binary state;

(5) a plurality of address registers for storing addresses, at least one of the address registers being assignable to store either a memory address corresponding to a memory location in the portion of the memory or a peripheral device address,

CHARACTERIZED IN THAT

BUREAU OMPI WIPO INTERNATIONAL

the op-code extension register specifies the assignment of each assignable address register when an instruction being executed by the central processing unit specifies the transfer of the contents of a particular address register to the address output;

the controller responsive to the contents of the op-code extension register providing a first binary state at the first output and a second binary state at the second output if the particular address register is assigned to store a peripheral device address; and

the controller providing a second binary state at the first output and a first binary state at the second output if the particular address register is assigned to store a memory address corresponding to a memory location in the portion of the memory.

BUREAU OMPI WIPO INTERNATIONAL

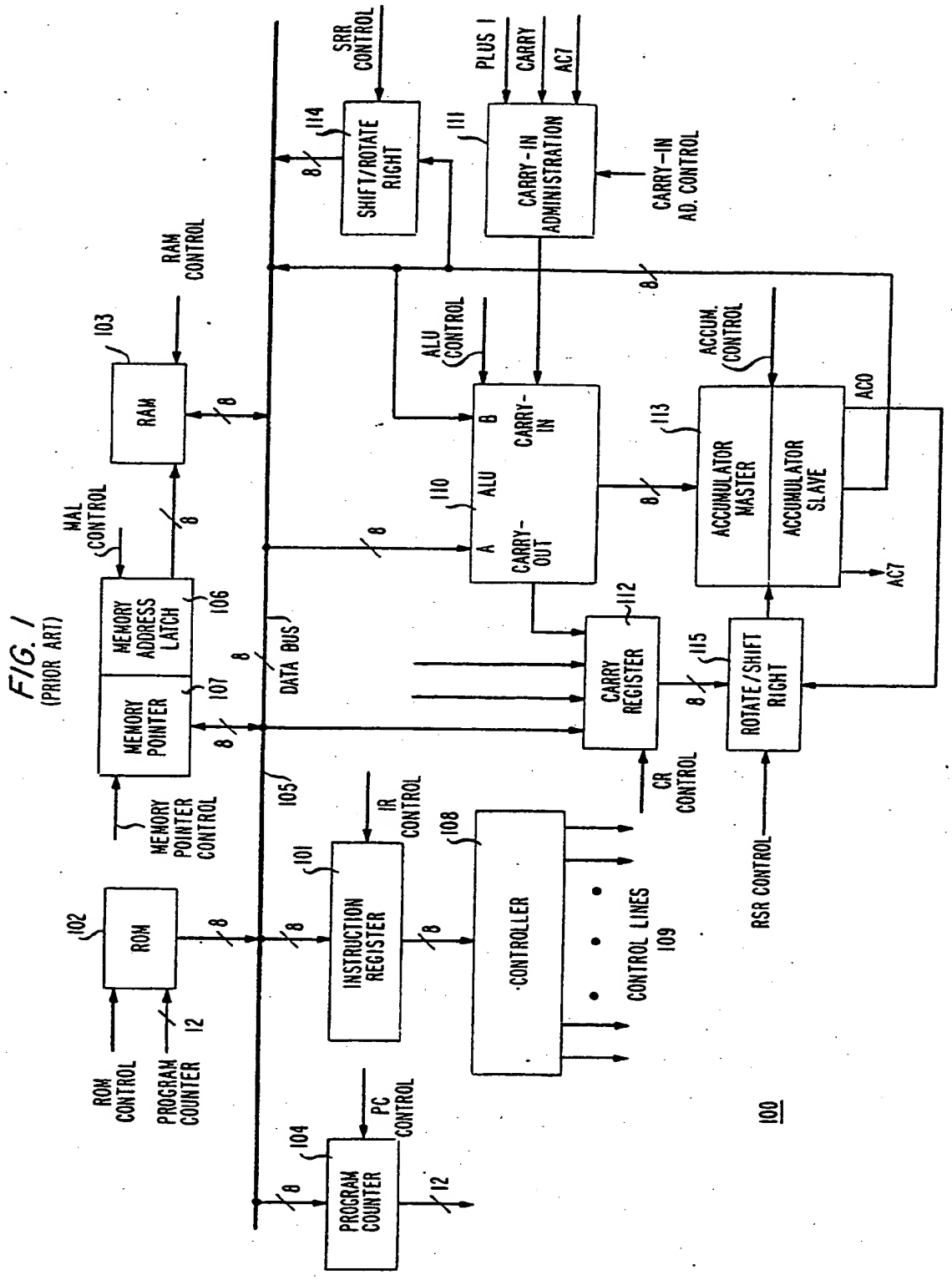


FIG. 1
(PRIOR ART)

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

FIG. 2

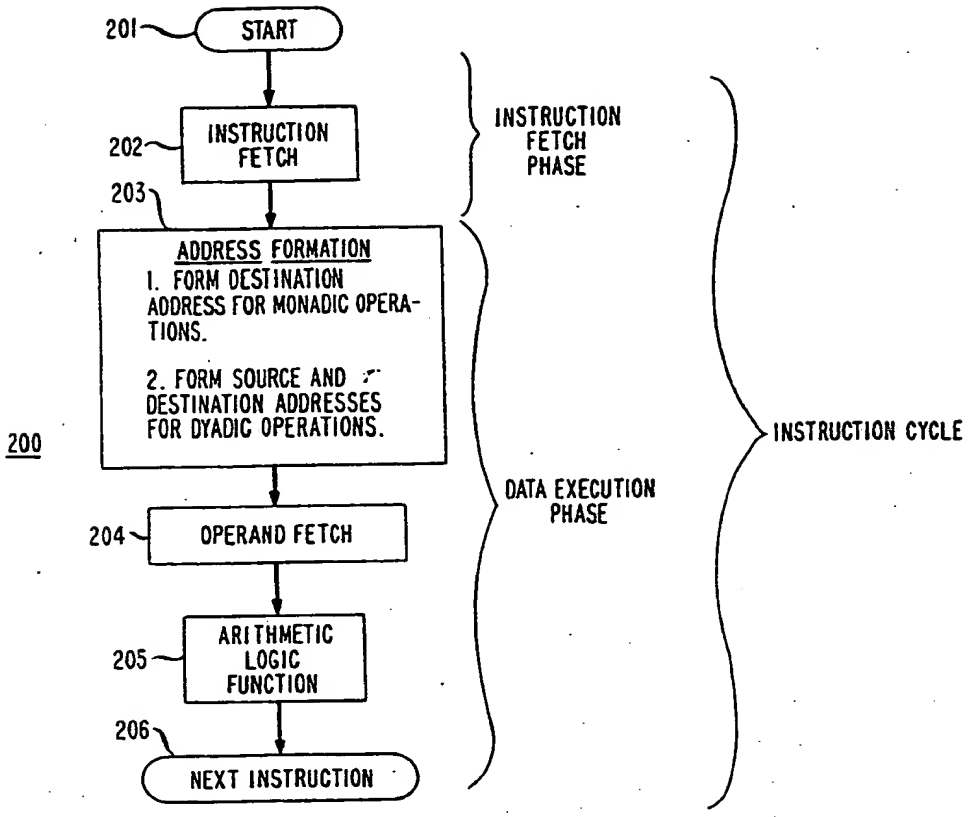
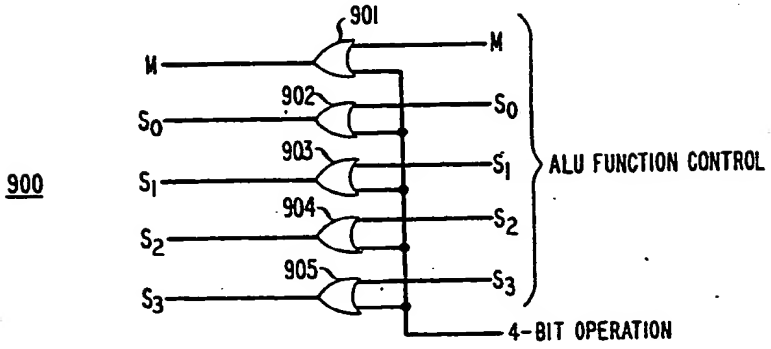


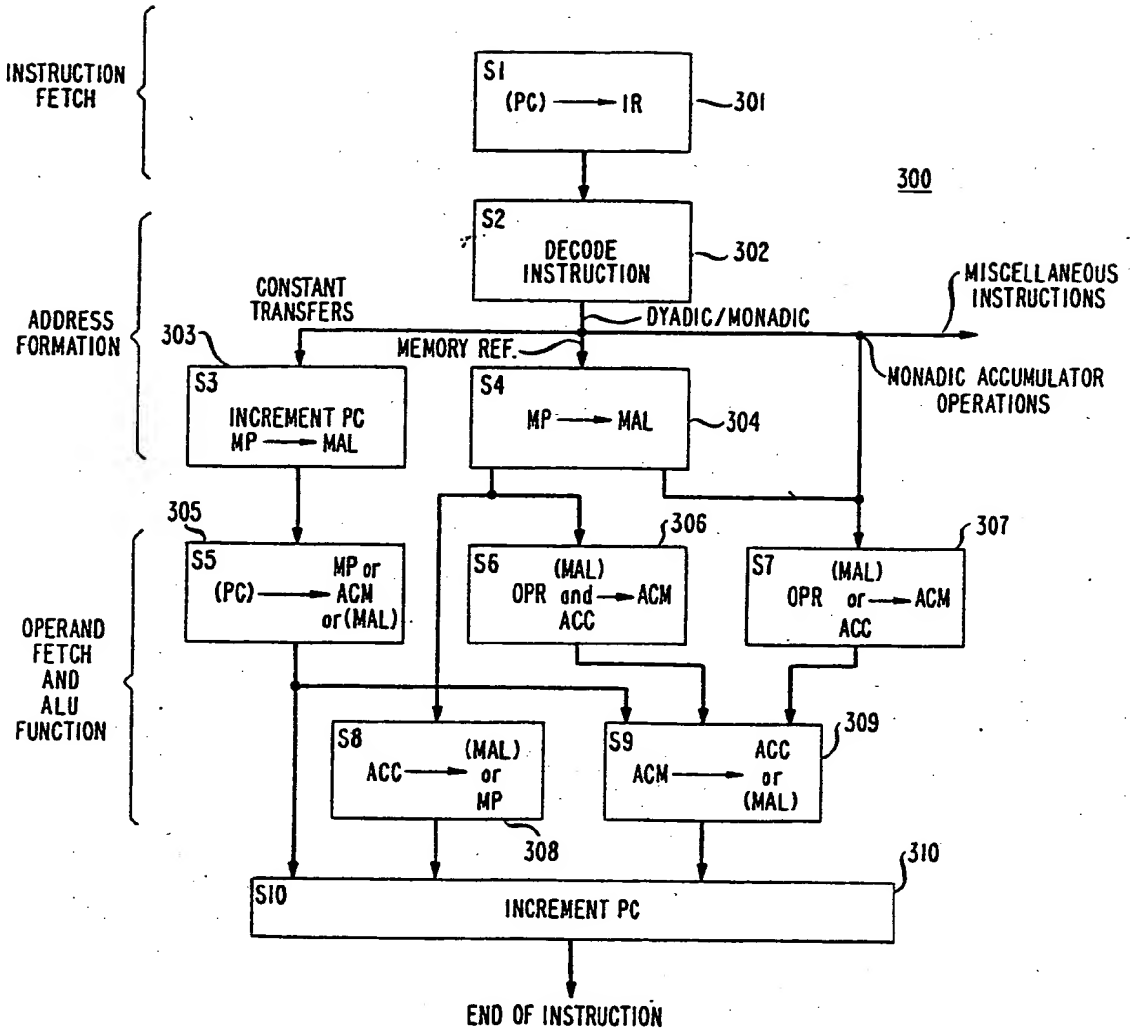
FIG. 9



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

FIG. 3



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

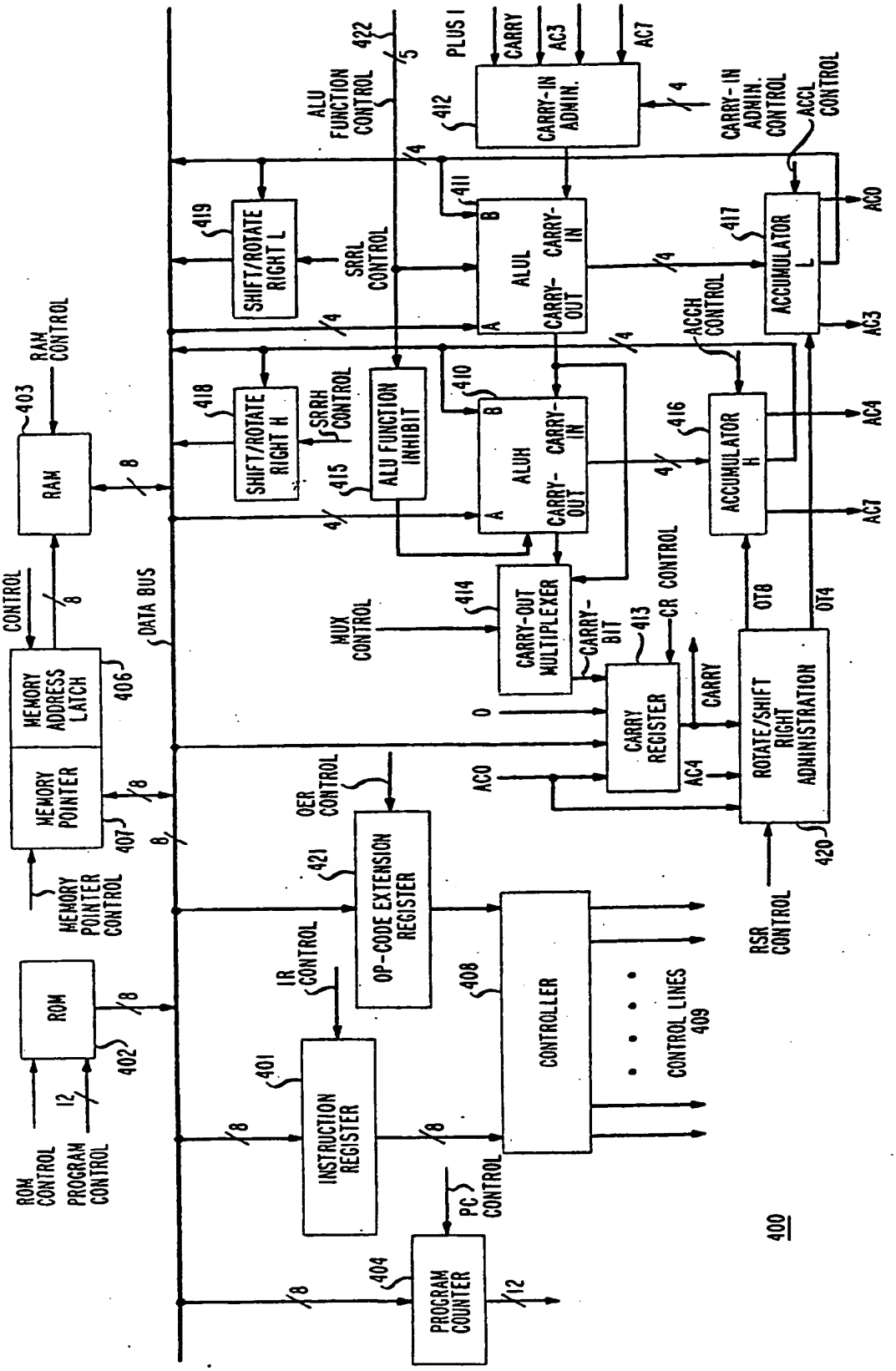


FIG. 4

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

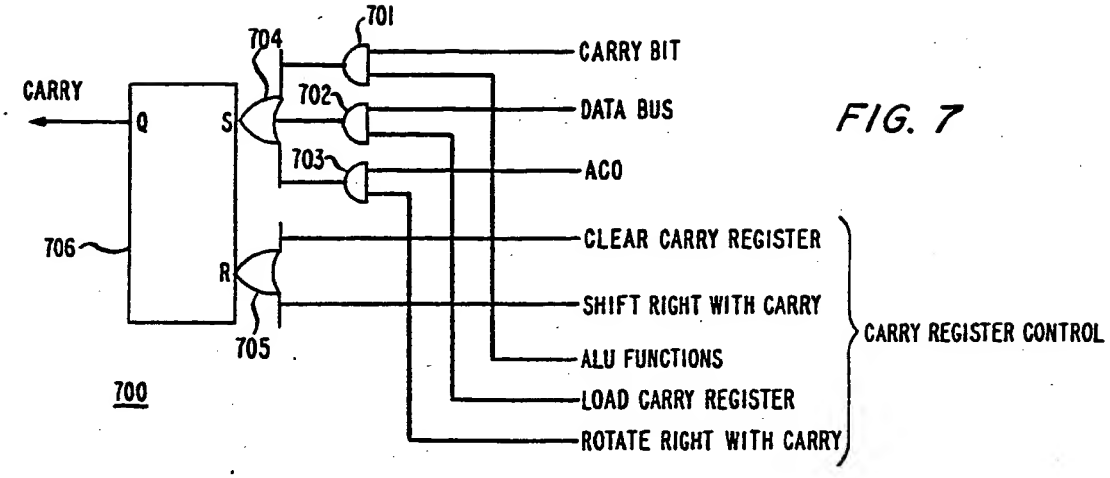


FIG. 7

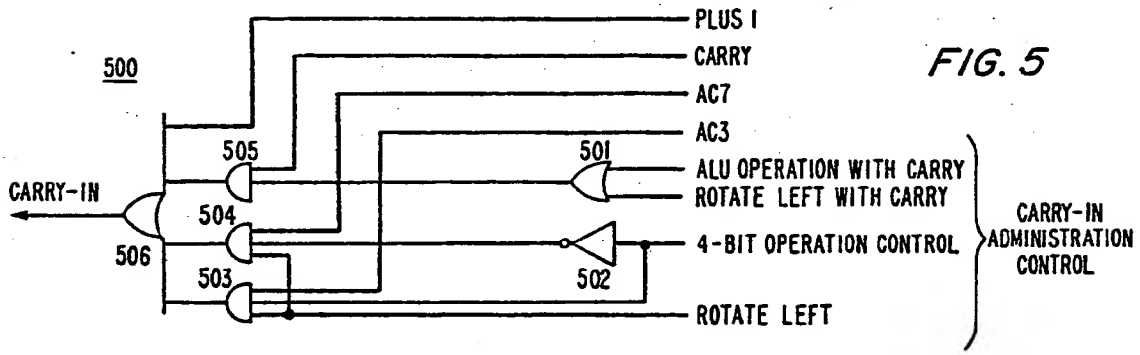


FIG. 5

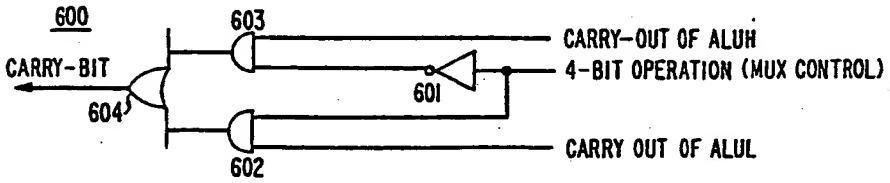


FIG. 6

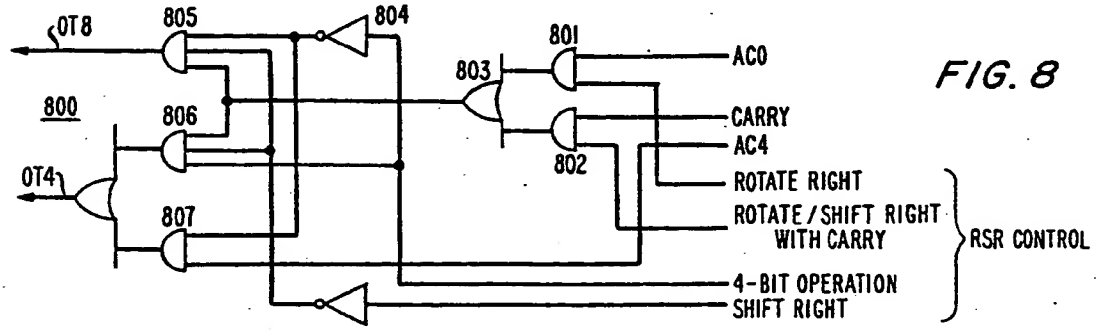


FIG. 8

BUREAU OMPI WIPO

BNSDOCID: <WO___8001423A1_I_>

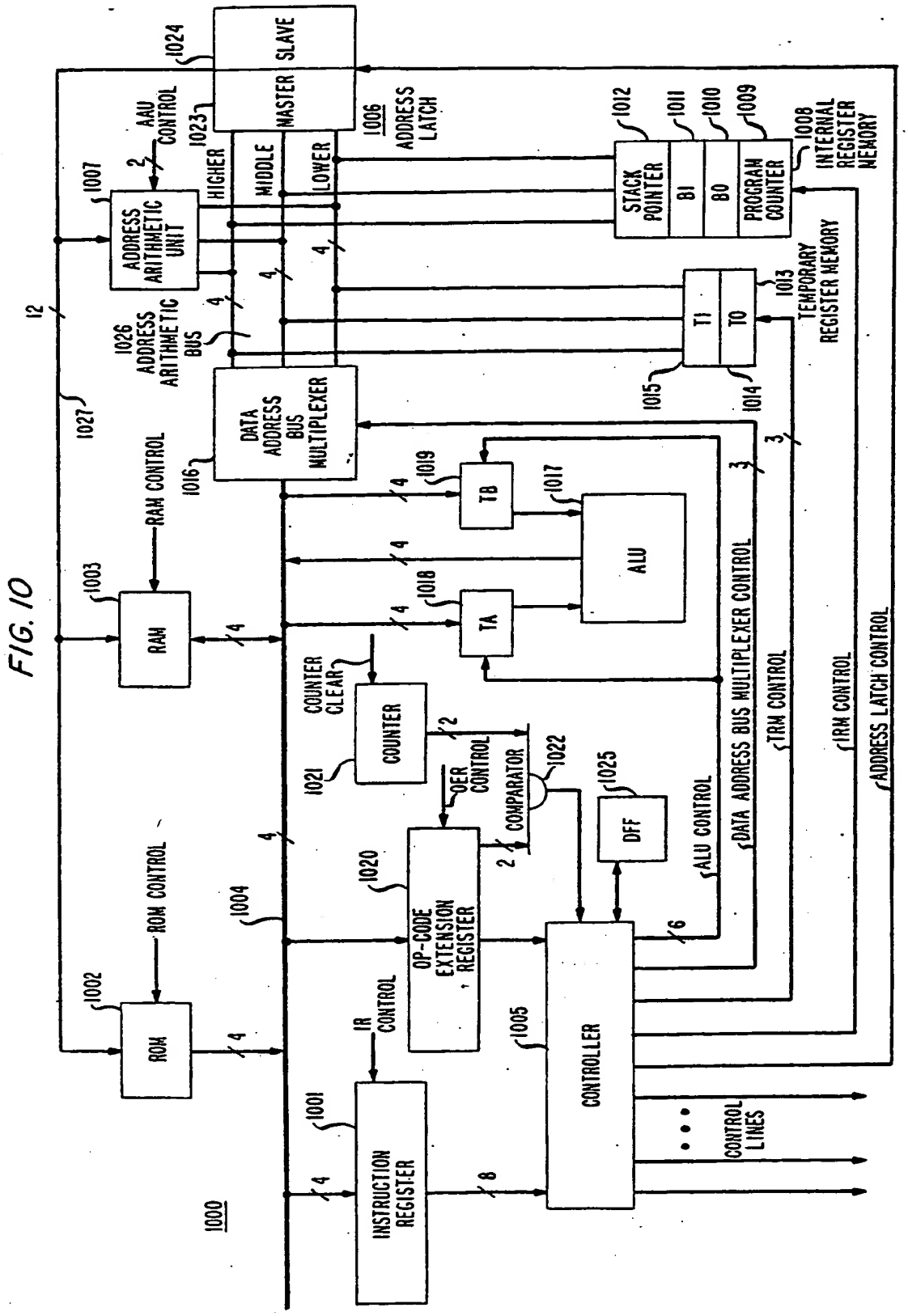


FIG. 10

BUREAU OMPI WIPO INTERNATIONAL

FIG. 14

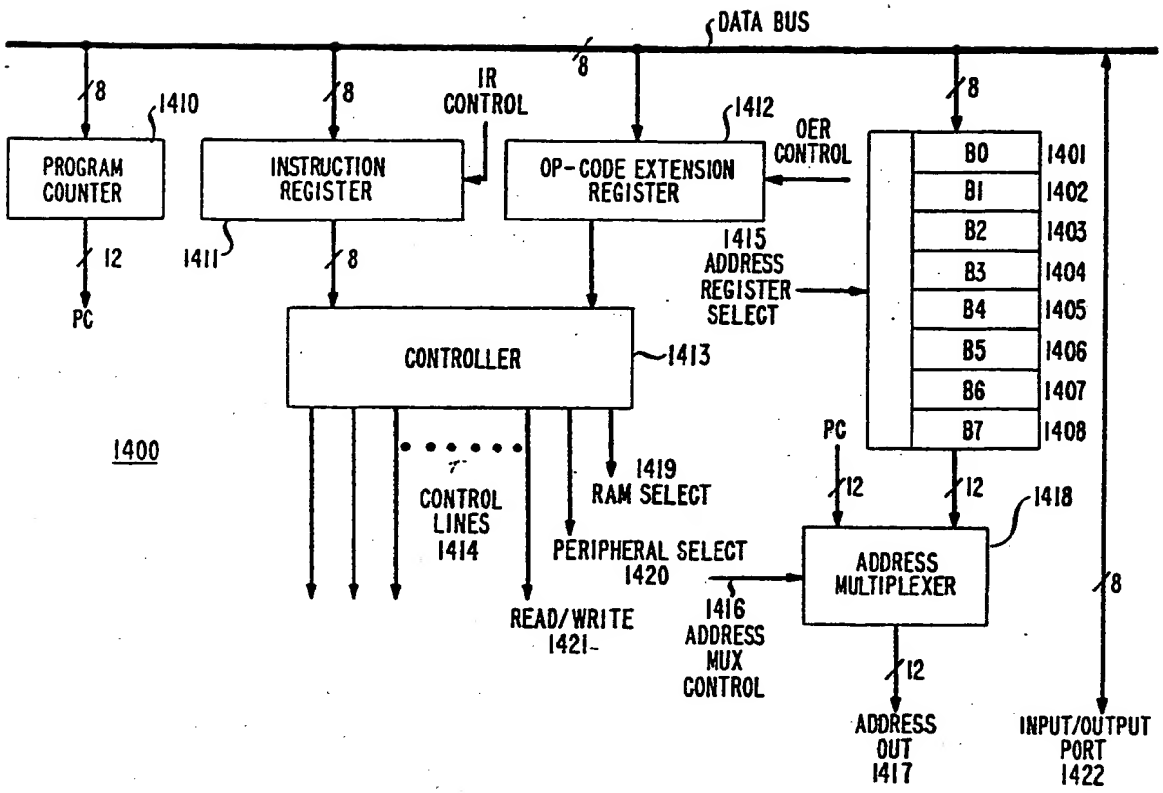
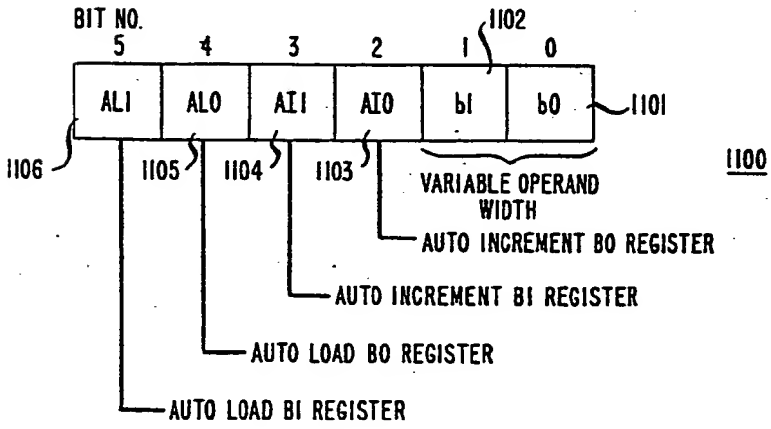


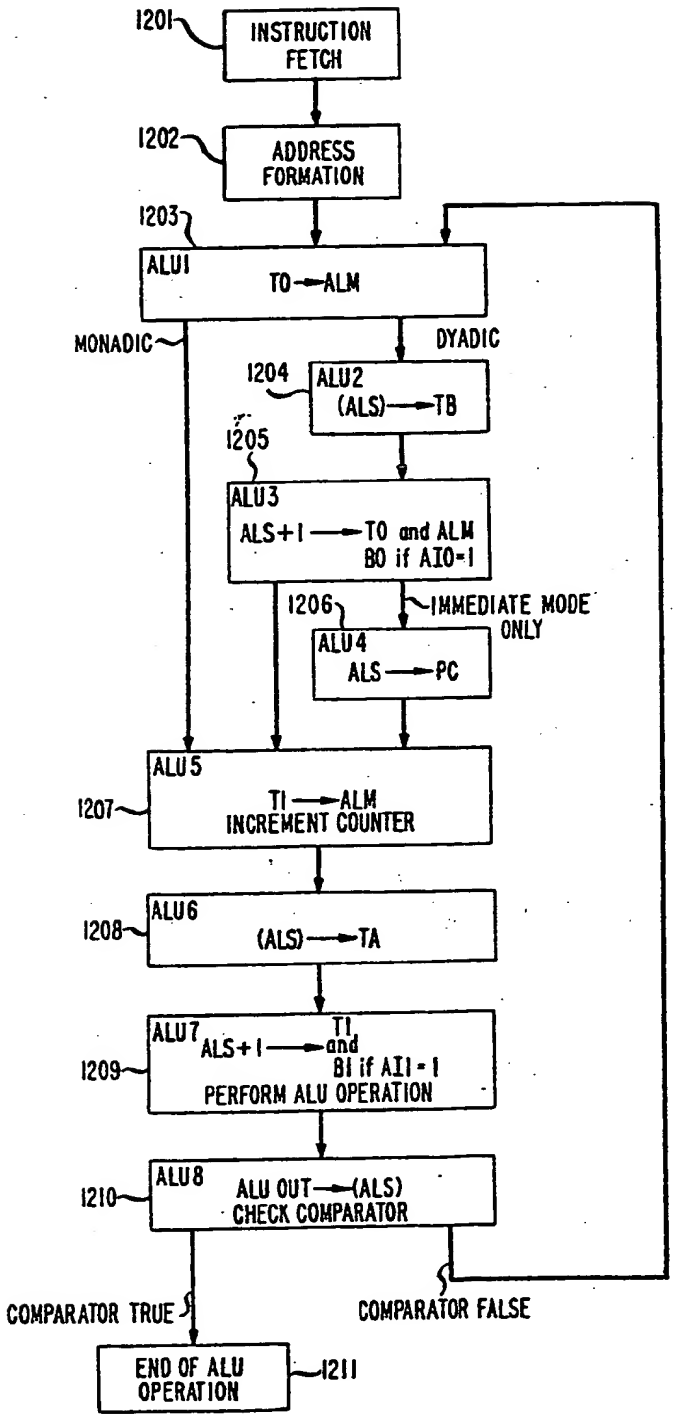
FIG. 11

OP-CODE EXTENSION REGISTER BIT ASSIGNMENT:



BUREAU OMPI

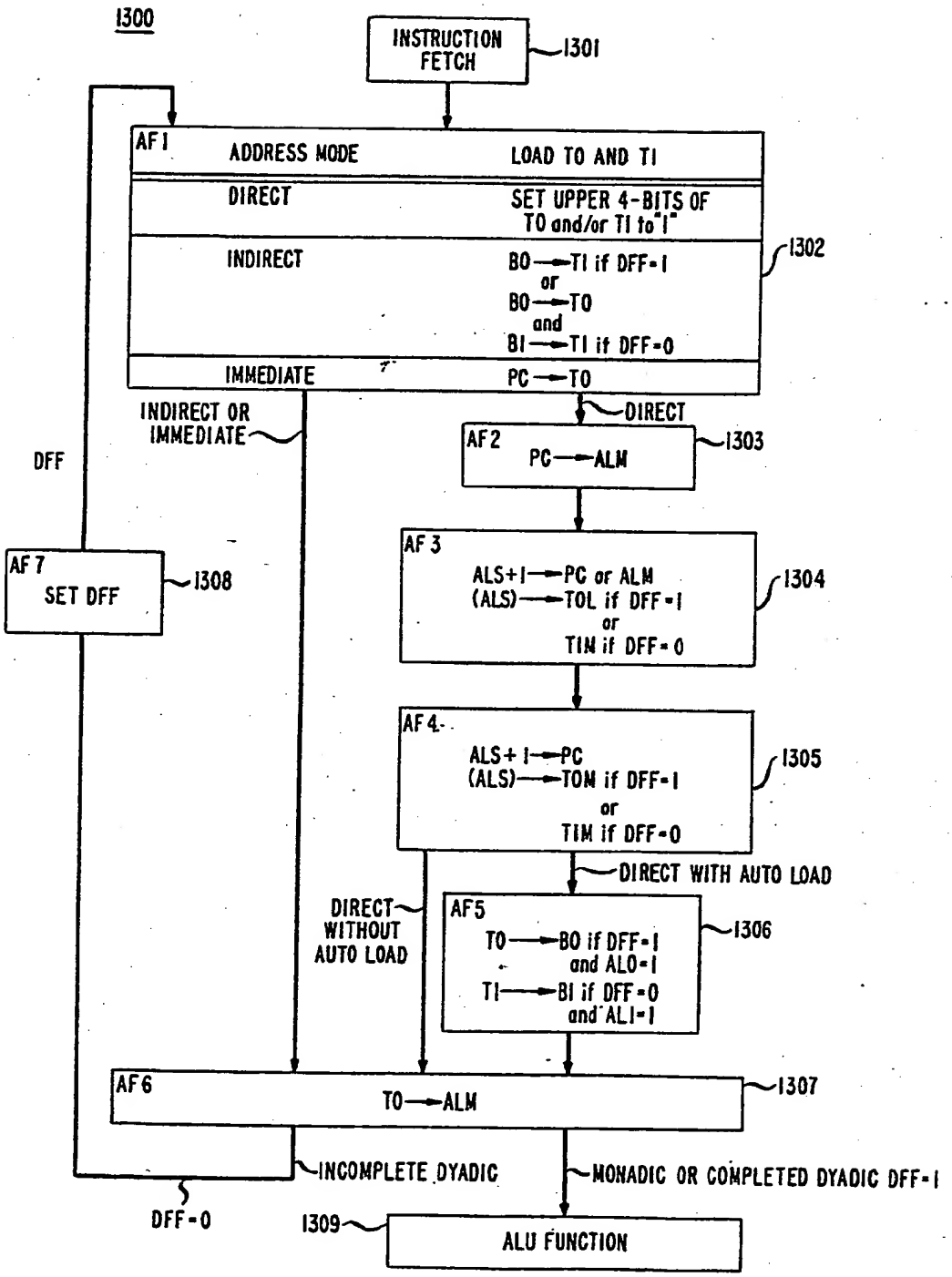
FIG. 12



BUREAU OMPI WIPO

BNSDOCID: <WO___8001423A1_I_>

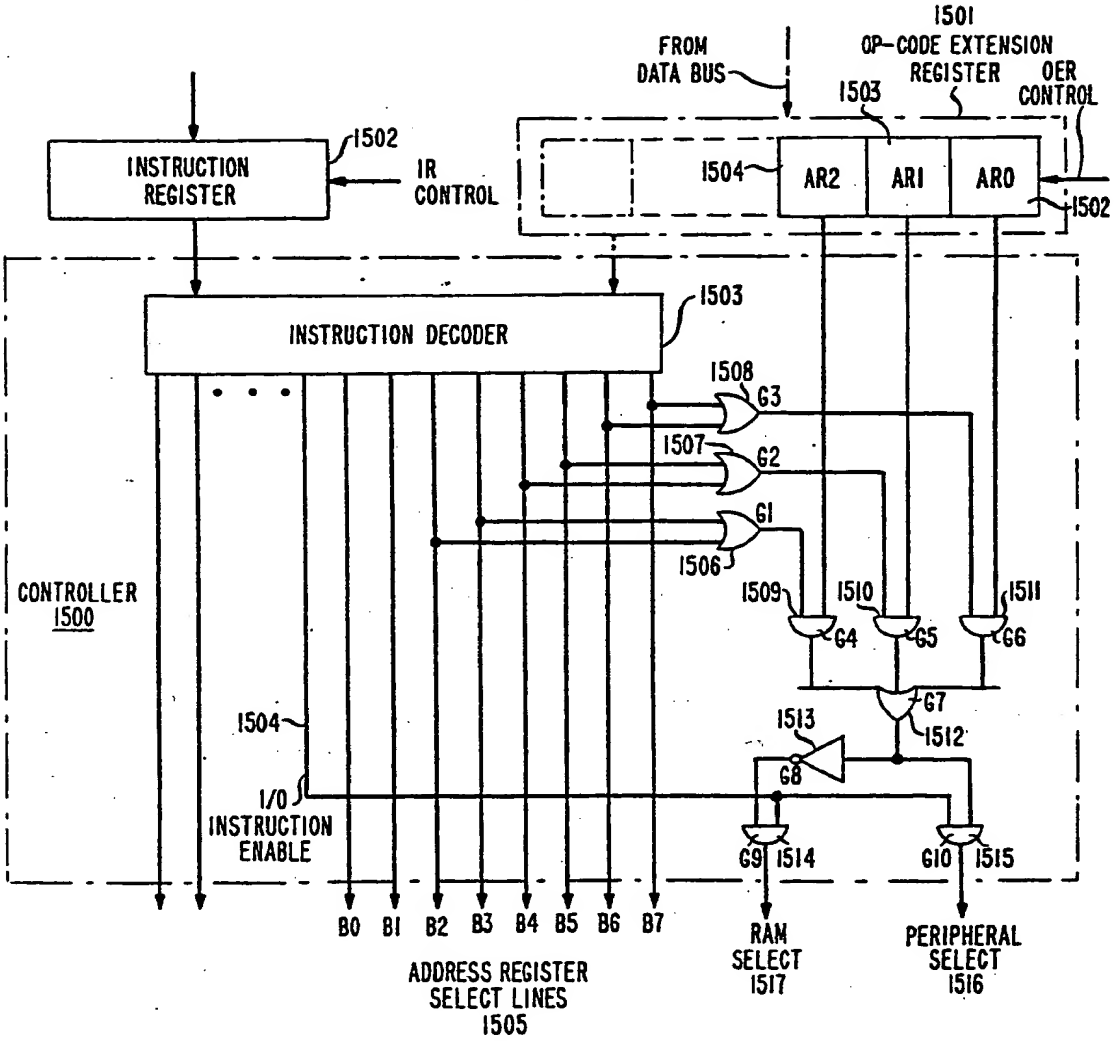
FIG. 13



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

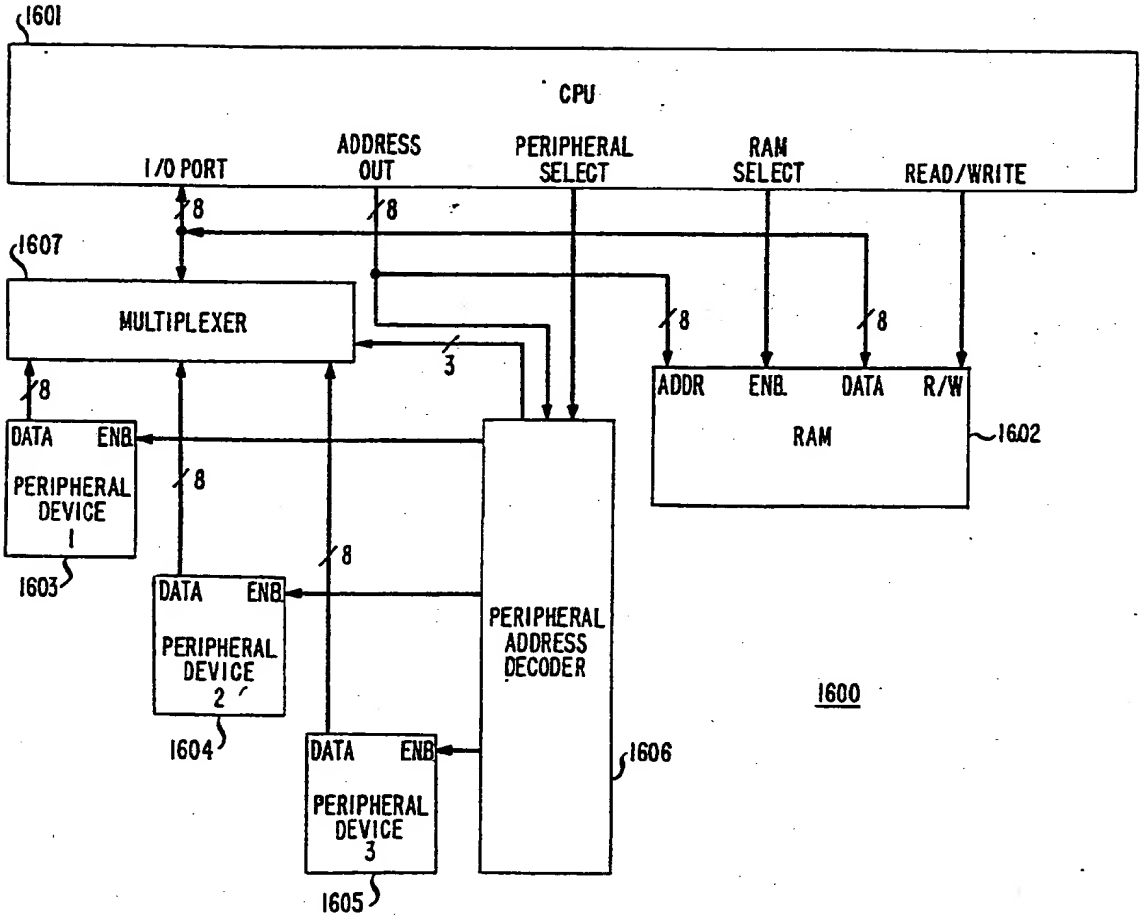
FIG. 15



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

FIG. 16



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1_I_>

INTERNATIONAL SEARCH REPORT

International Application No PCT/US79/01045

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [3]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int. Cl[3] G 06F 7/38, G 11C 8/00

U. S. Cl. 364/200, 900, 748

WO 80/01423

II. FIELDS SEARCHED

Minimum Documentation Searched [4]

Classification System	Classification Symbols
US	364/200,900,748

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [5]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [14]

Category *	Citation of Document, [16] with indication, where appropriate, of the relevant passages [17]	Relevant to Claim No. [18]
X	N, IBM Technical Disclosure Bulletin, vol. 8 No. 12, Published 12, May 1966, J. Dirac, Call Instruction, page 1751	1,5
X	US, A, 4,117,536, Published 26 September 1978	1,5
A	US, A, 4,021,655, Published 03, May 1977	2
X	US, A, 3,593,312, Published 13, July 1971	2,4
A	US, A, 3,234,370, Published 08 February 1966	2,3,4
A P	US, A, 4,161,784, Published 17, July 1979	2,3
A	N, MOS Technology, Inc. Publication number 6500-50A, published January 1976, MCS6500 Microcomputer Family Programming Manual, Page 23-30	1,5

* Special categories of cited documents: [15]

"A" document defining the general state of the art

"E" earlier document but published on or after the international filing date

"L" document cited for special reason other than those referred to in the other categories

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but on or after the priority date claimed

"T" later document published on or after the international filing date or priority date and not in conflict with the application, but cited to understand the principle or theory underlying the invention

"X" document of particular relevance

IV. CERTIFICATION

Date of the Actual Completion of the International Search [2]	Date of Mailing of this International Search Report [2]
17, April 1980	25 APR 1980
International Searching Authority [1]	**Signature of Authorized Officer [20]**
ISA/US	David Eng David Eng

Form PCT/ISA/210 (second sheet) (October 1977)

BNSDOCID: <WO__8001423A1_I_>

This Page Blank (uspto)

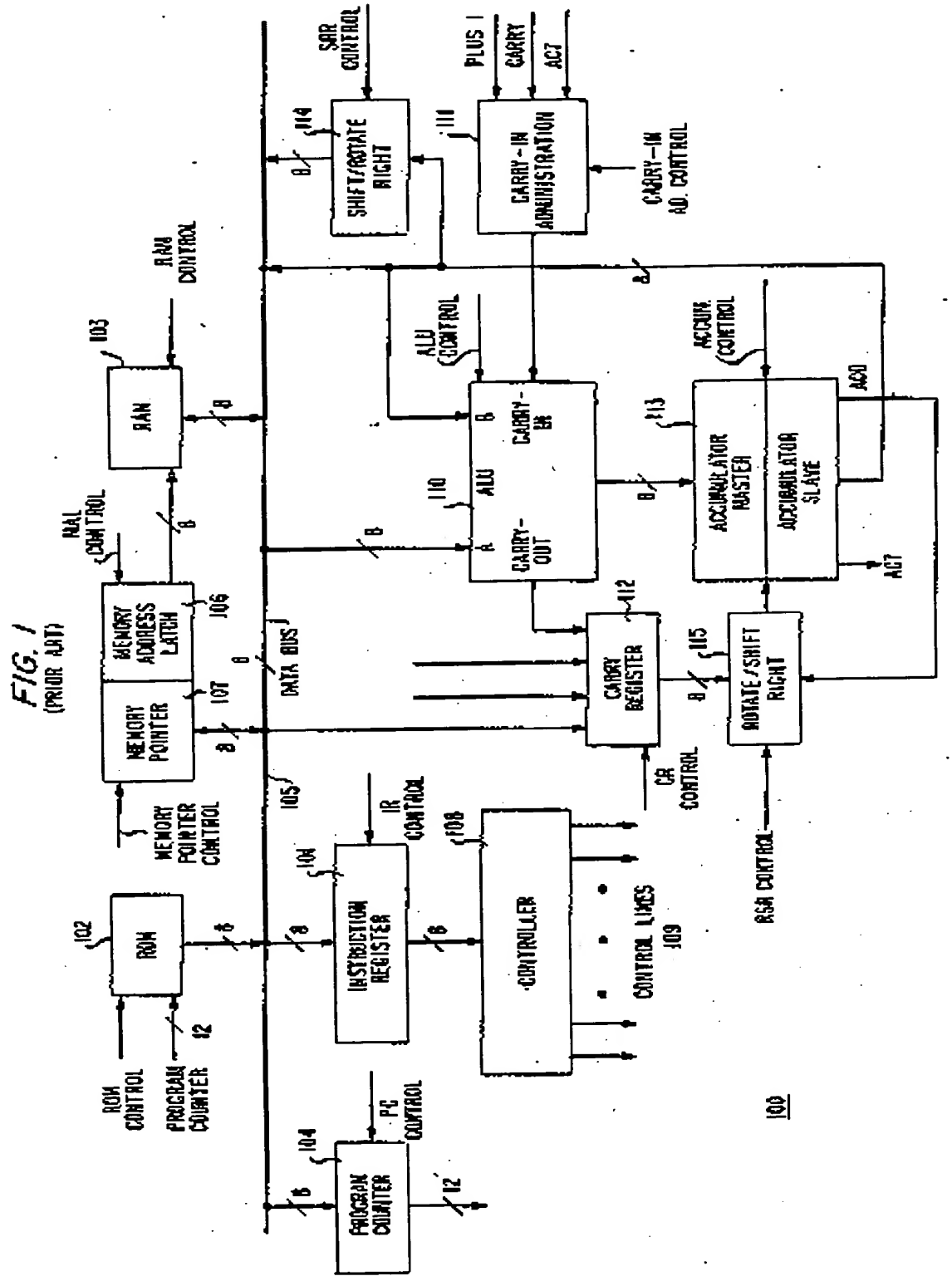


FIG. 1
(PRIOR ART)

BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1TI_>

FIG. 2

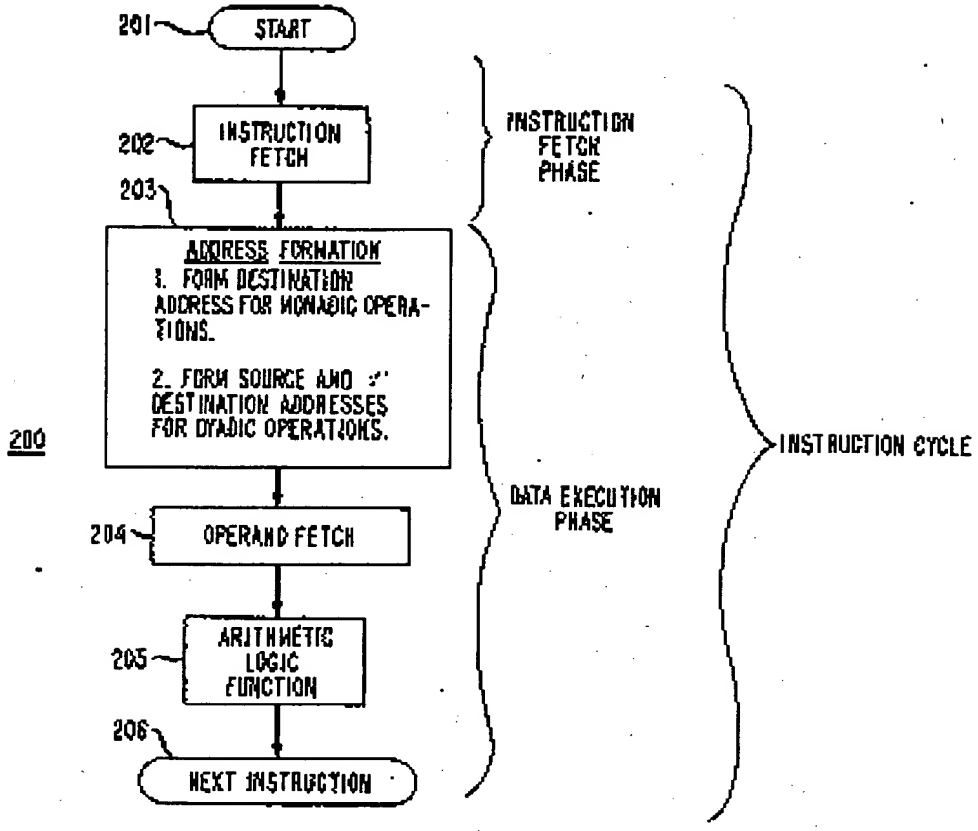
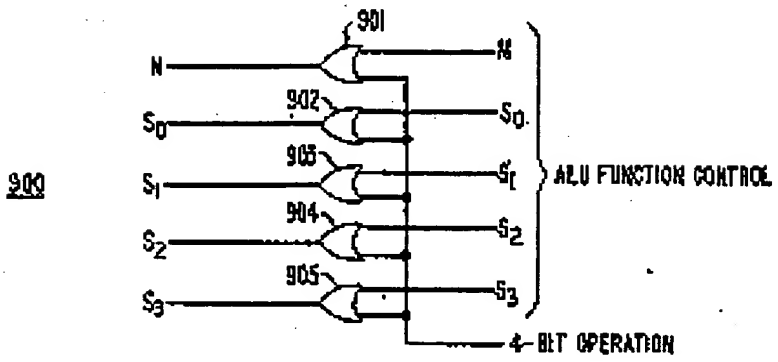


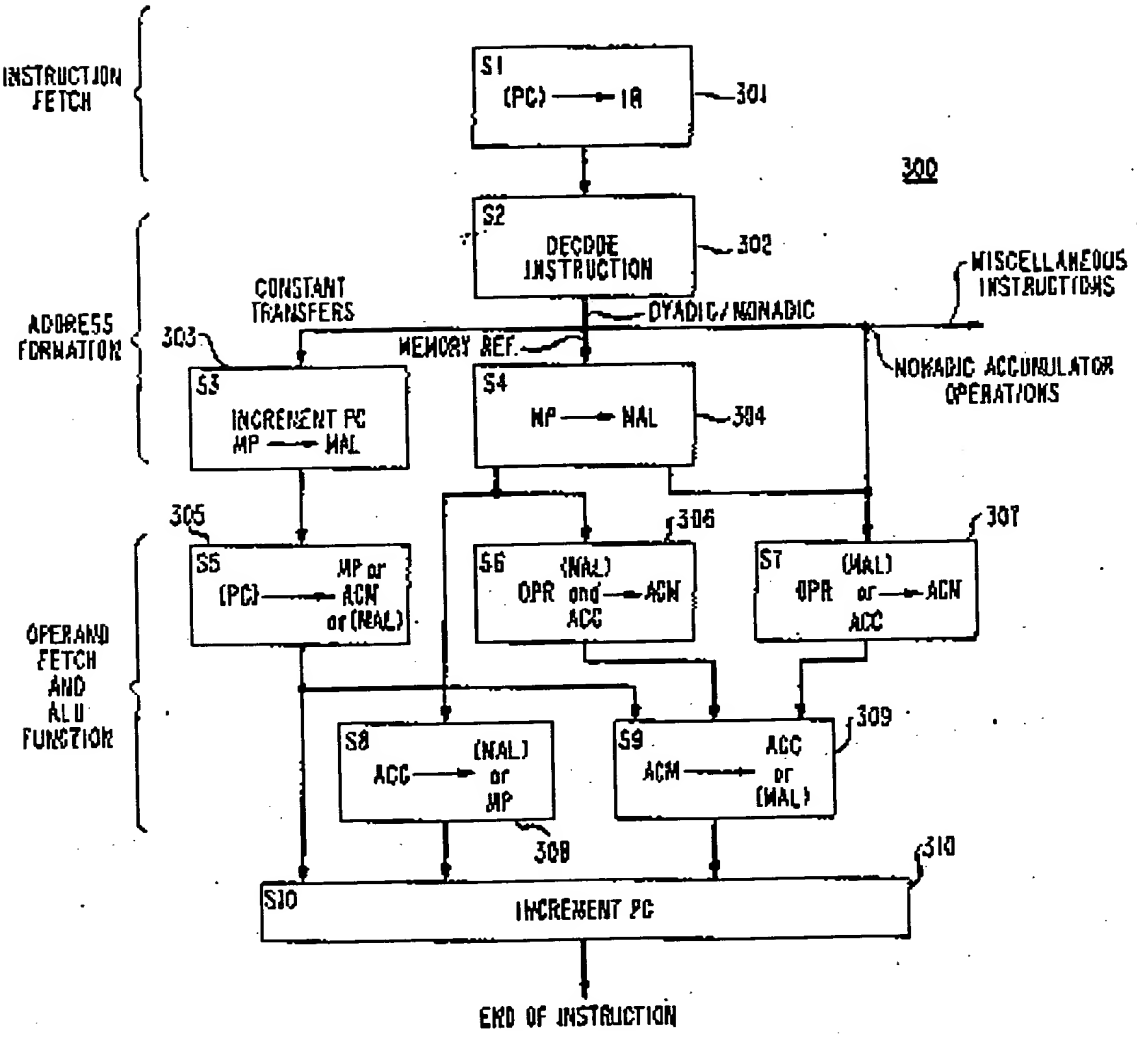
FIG. 9



BUREAU OMPI WIPO INTERNATIONAL

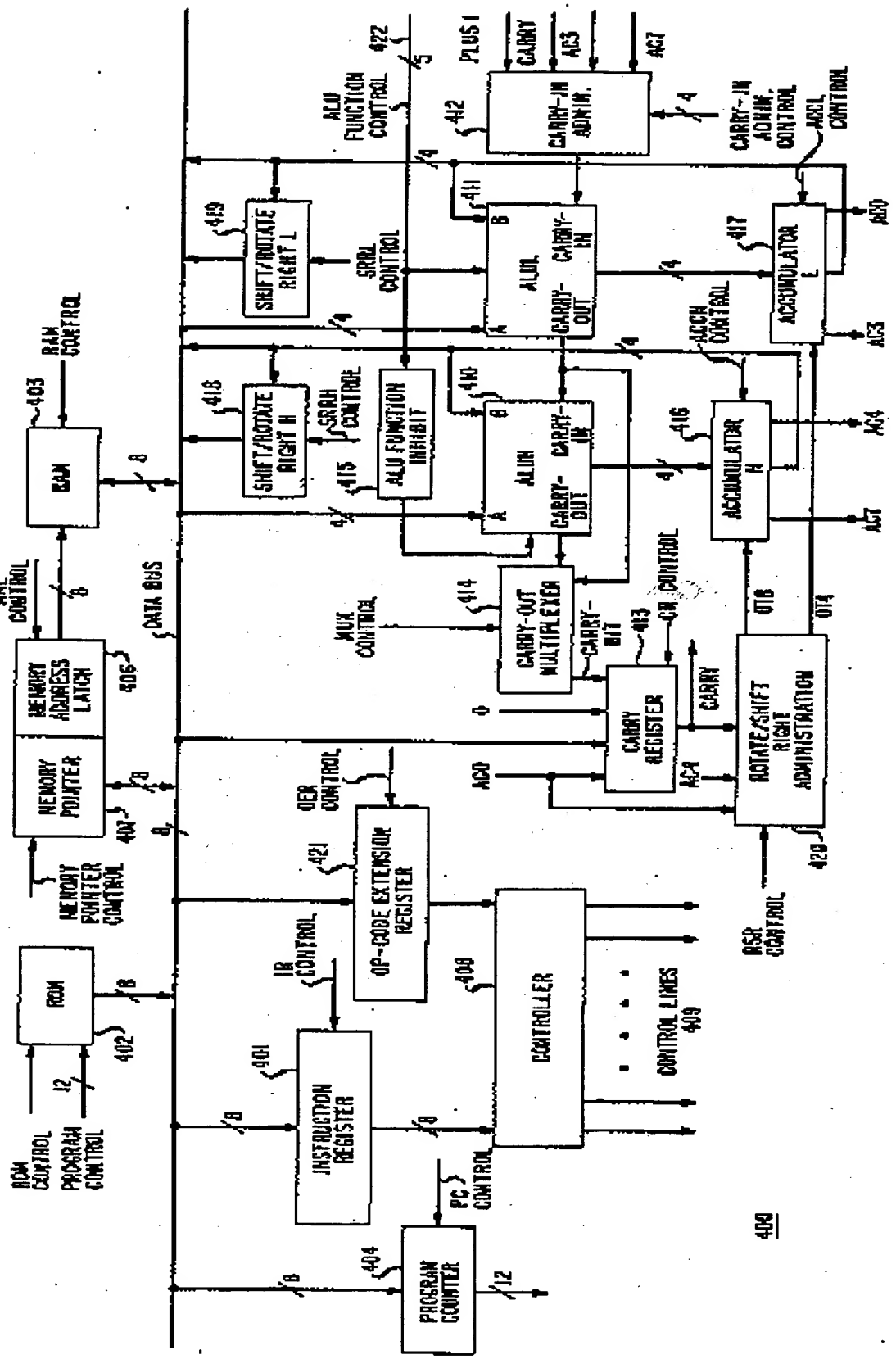
BNSDOCID: <WO___8001423A1TI_>

FIG. 3



BUREAU OMPI WIPO INTERNATIONAL

FIG 4



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1TI_>

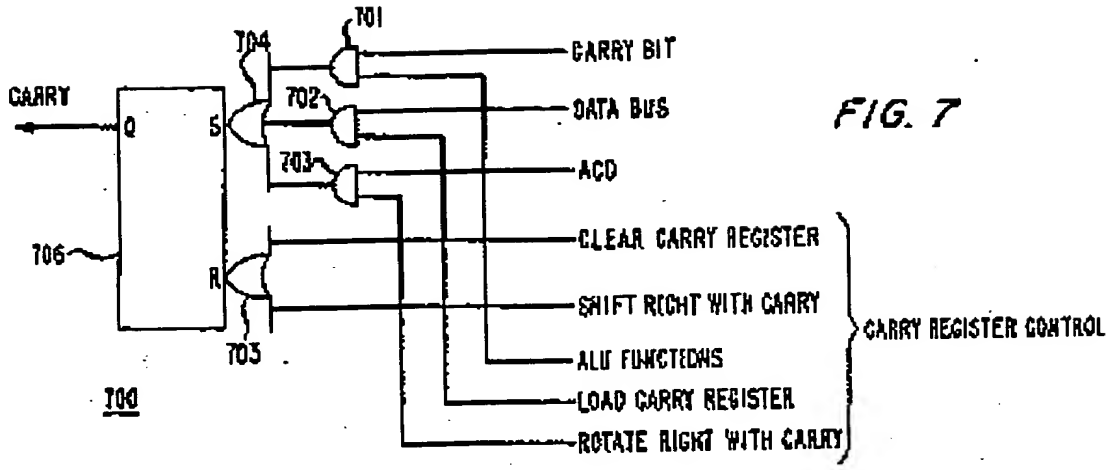


FIG. 7

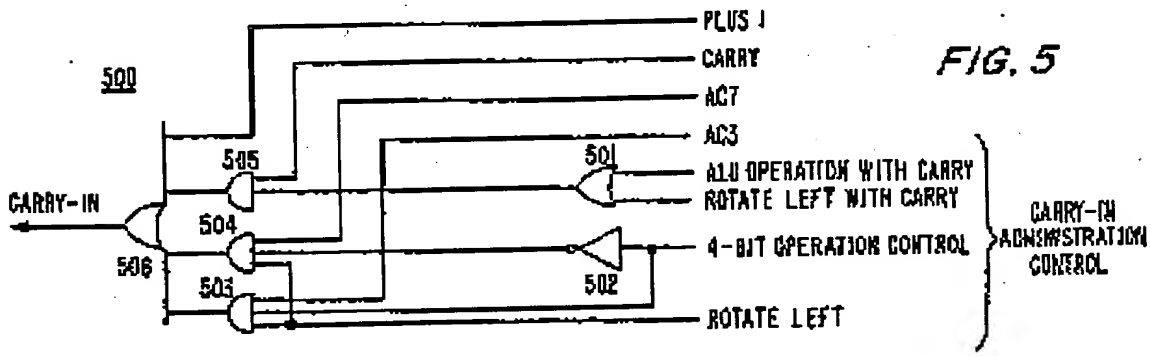


FIG. 5

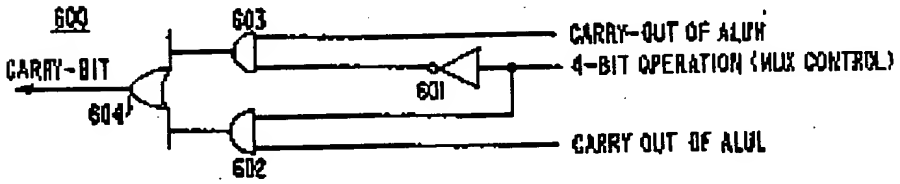


FIG. 6

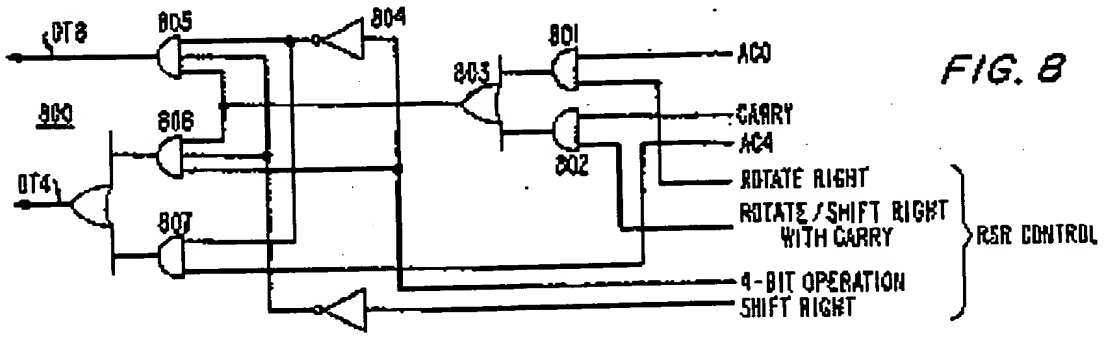
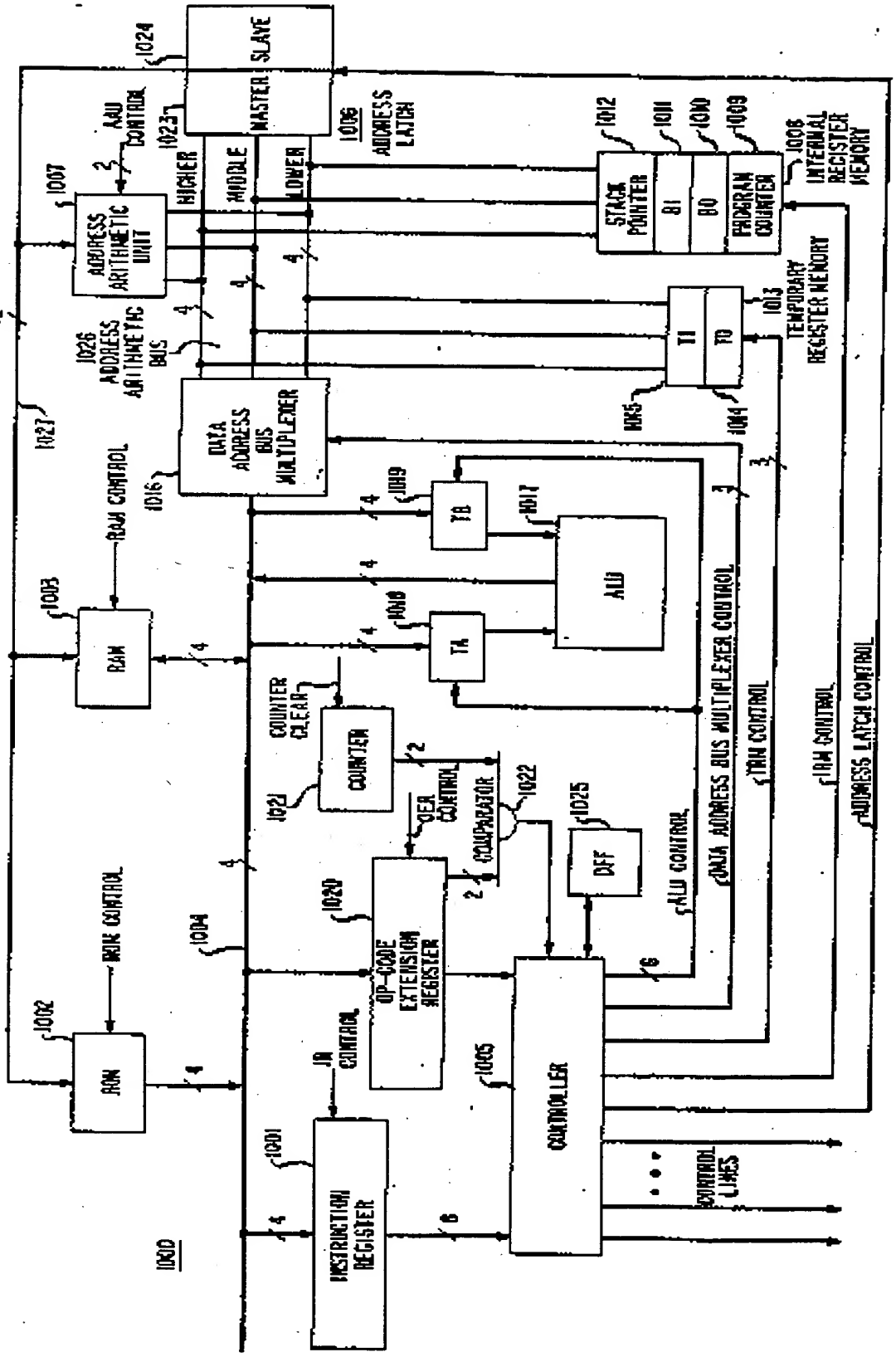


FIG. 8

BUREAU OMPI WIPO

FIG. 10



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1TI_>

FIG. 14

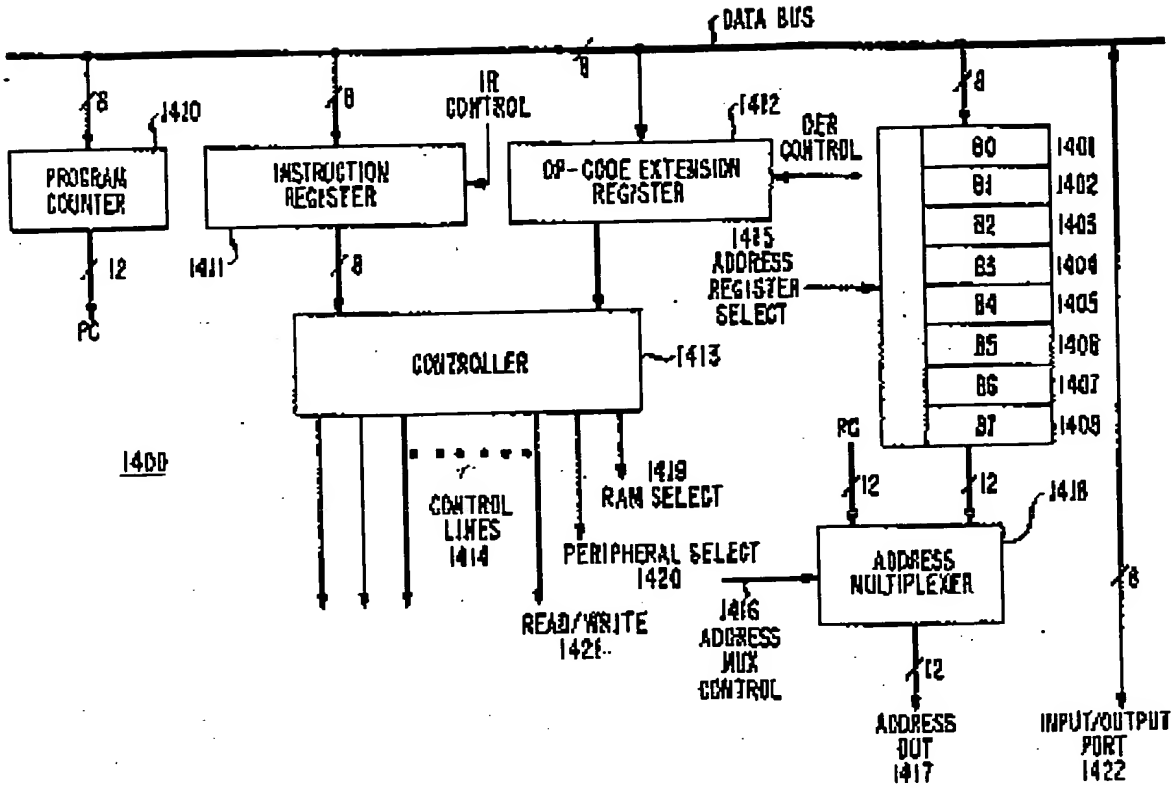
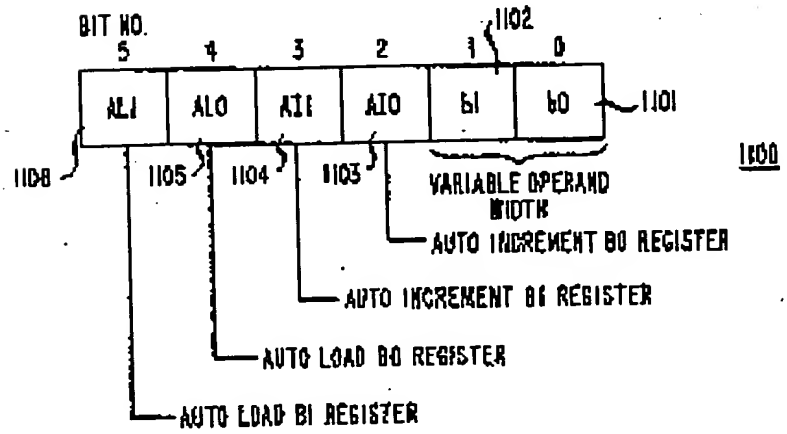


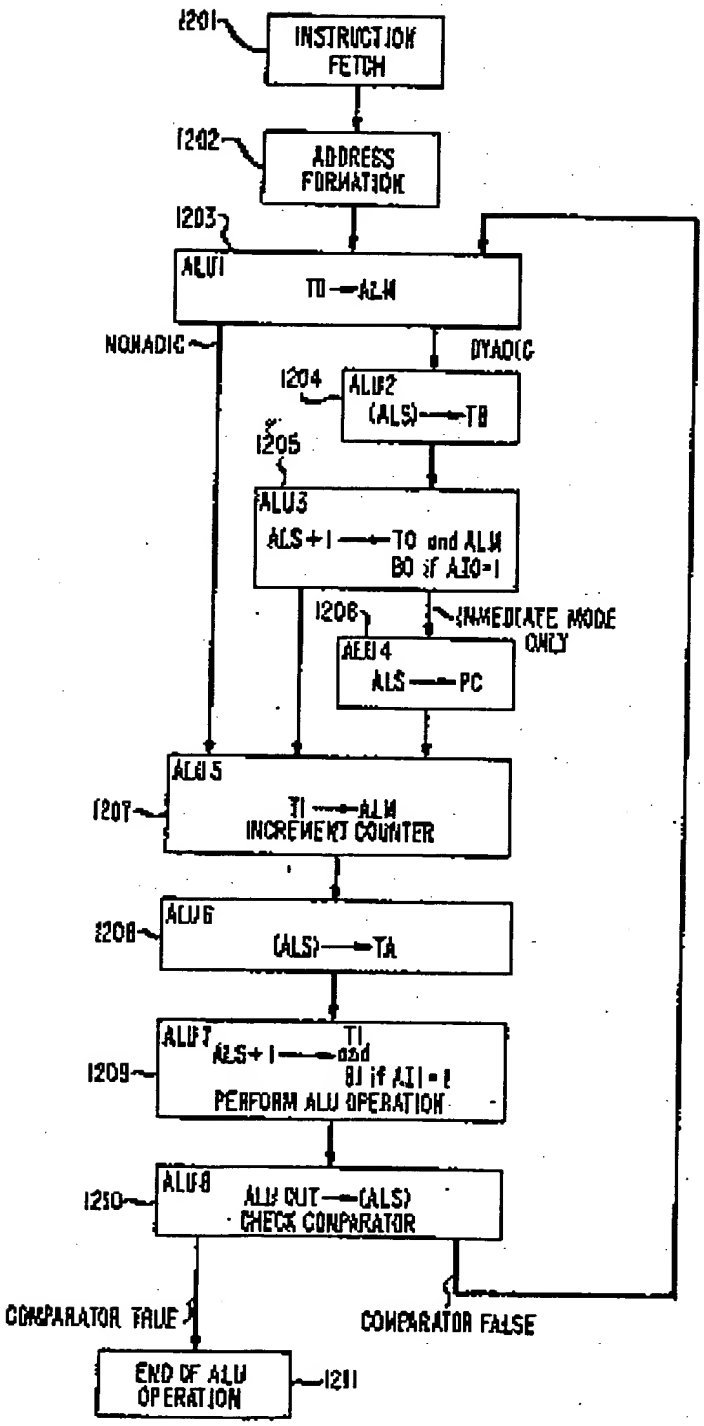
FIG. 11

OP-CODE EXTENSION REGISTER BIT ASSIGNMENT:



BUREAU OMPI

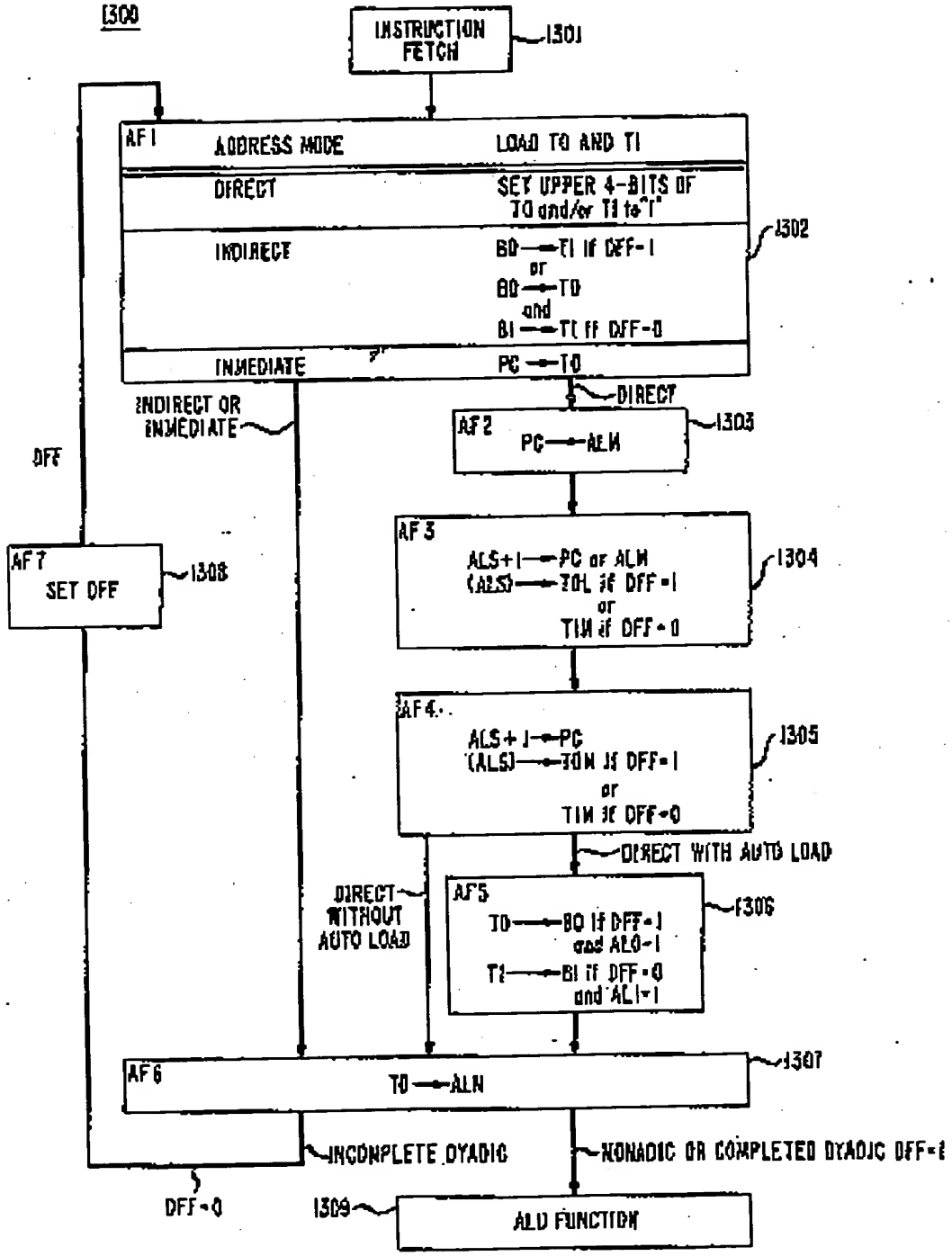
FIG. 12



BUREAU OMPI WIPO

BNSDOCID: <WO___8001423A1TI_>

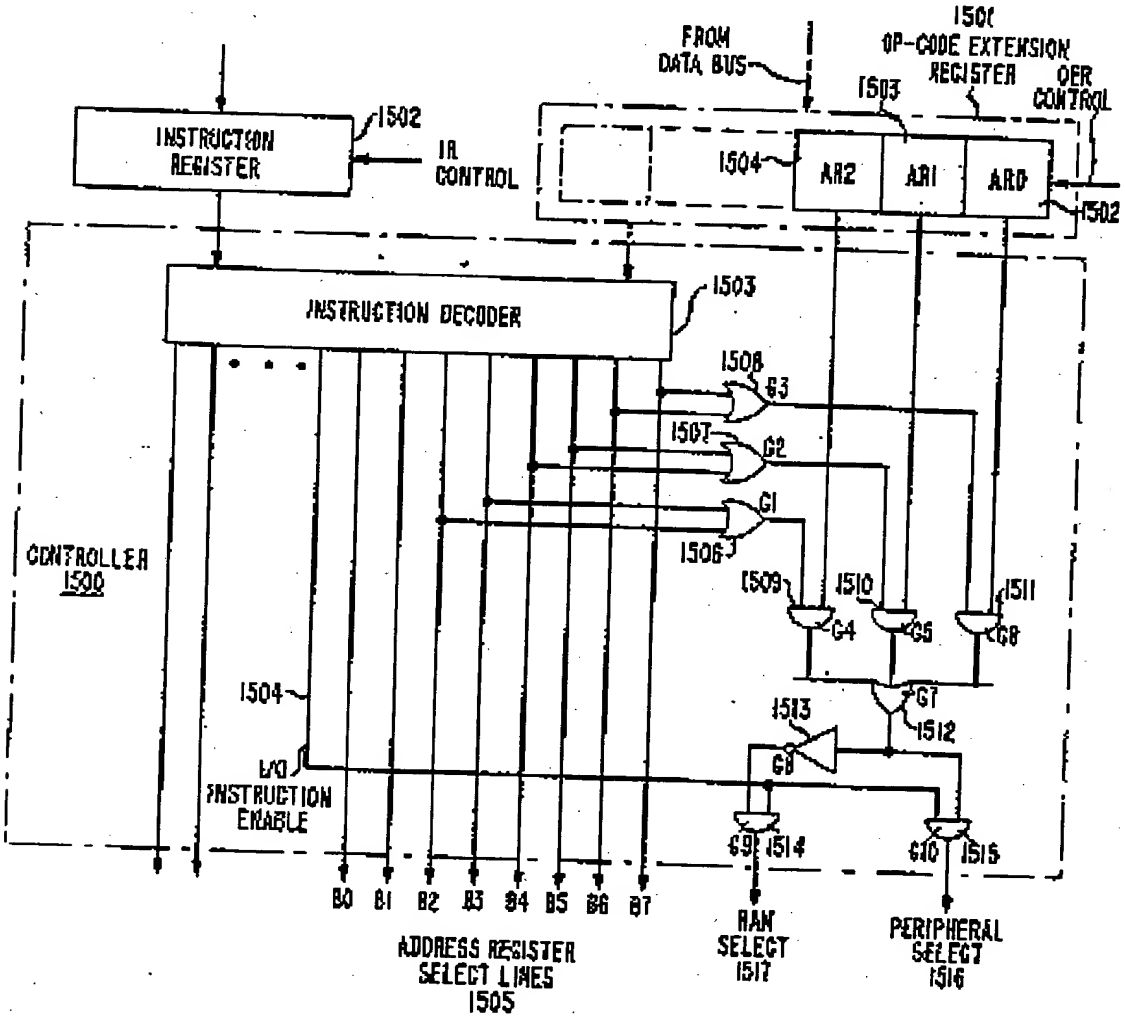
FIG. 13



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO__8001423A1T1_>

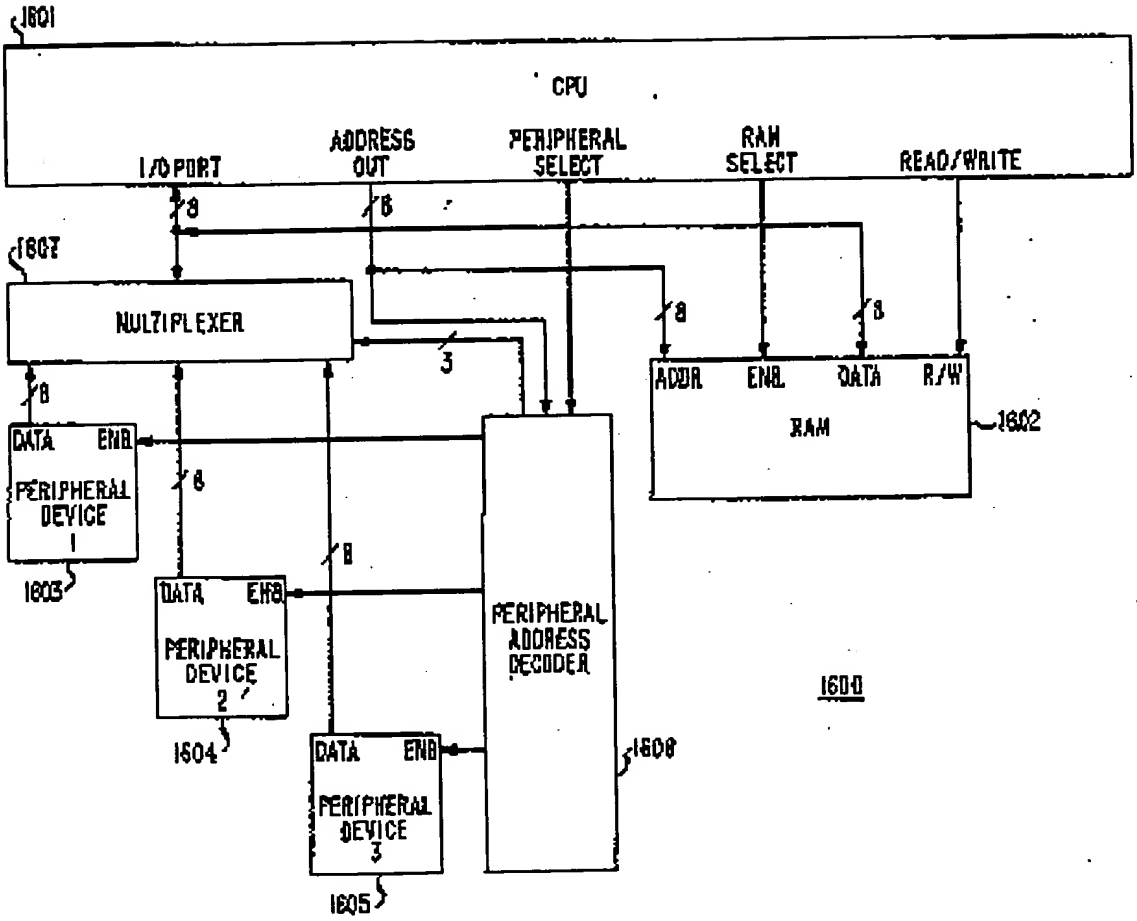
FIG. 15



BNSDOCID: <WO___8001423A1TI_>

BUREAU OMPI WIPO INTERNATIONAL

FIG. 16



BUREAU OMPI WIPO INTERNATIONAL

BNSDOCID: <WO___8001423A1TI_>

This Page Blank (uspto)